Departamento de Informática
COmputer SECurity Lab (COSEC)

INGENIERÍA EN INFORMÁTICA

PROYECTO FIN DE CARRERA

Desarrollo de un método de autenticación de emergencia basado en el cuarto factor: *"Alguien a quien el usuario conoce"*

Autora: Elena Frau Carro
Tutores: José María de Fuentes García-Romero de Tejada
Lorena González Manzano

Leganés, marzo de 2015

Título: Desarrollo de un método de autenticación de emergencia basado en el cuarto factor: "Alguien a quien el usuario conoce"
Autor: Elena Frau Carro
Tutores: José María de Fuentes García-Romero de Tejada
Lorena González Manzano

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día _24_ de _Marzo_ de _2015_ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

El ritmo a veces desenfrenado del día a día hace que muchas veces no nos paremos a reflexionar y a dar las gracias a las personas de nuestro entorno por todo aquello que nos dan y hacen por nosotros. Por eso hoy, no quiero dejar pasar la oportunidad de dedicar unas palabras de agradecimiento a aquellas personas sin las cuales llegar hasta aquí habría sido imposible.

En primer lugar quiero agradecer a mis padres las oportunidades, que con su esfuerzo, me han brindado en la vida. Entre ellas está la de haber podido realizar esta carrera, gracias a la cual hoy en día puedo decir que vivo de una profesión que me llena y me permite crecer. Gracias también a ellos y a mi hermana por haber estado a mi lado apoyándome durante este largo camino, que hoy finaliza con la lectura de este proyecto. Porque a pesar del tiempo que me ha llevado finalizar esta etapa, nunca han perdido la confianza en que podía hacerlo y me han apoyado hasta el final.

También quiero agradecer a Marcos su infinita paciencia, su apoyo incondicional y el haber estado a mi lado en los buenos y en los malos momentos. He de decir que gracias a sus palabras de ánimo estoy hoy aquí.

Por último, pero no por ello menos importante, envío mis más sinceros agradecimientos a mis tutores y amigos, Chema y Lorena. Por su comprensión, por el positivismo que contagian, y por supuesto por la dedicación y profesionalidad con las que se han entregado en la tutela y dirección de este proyecto.

¡GRACIAS!

Resumen

La autenticación basada en las relaciones sociales no es para nada una nueva forma de autenticar a las personas. Desde tiempos inmemoriales los humanos establecemos continuamente protocolos de identificación, e implícitamente de autenticación, cada vez que interactuamos con las personas de nuestro entorno. Este proceso es llevado a cabo con tal naturalidad que pasa totalmente desaparecido.

En el año 2006 un equipo de investigadores de RSA Laboratories proponen un nuevo factor de autenticación adicional a los tres factores clásicos: “algo que el usuario conoce”, “algo que el usuario tiene” y “algo que el usuario es”, al que denominan el cuarto factor: “Alguien a quien el usuario conoce”. Este factor basado en las relaciones sociales que los individuos establecen con las personas de su entorno es trasladado al ámbito de los sistemas informáticos como mediante el proceso de *vouching*.

El presente proyecto toma como punto de partida el cuarto factor y el concepto de *vouching* para desarrollar un sistema de autenticación de emergencia para el conocido gestor de contenidos Joomla!

Palabras clave: autenticación, voucher, factores de autenticación, Joomla

Abstract

Authentication based on social networks is not a new way of authenticating people at all. Since old ages, humans constantly establish identification and, implicitly, authentication protocols every time they interact with people around them. This process is carried out in such a natural way that goes totally unnoticed.

In 2006 a team of researchers from RSA Laboratories proposed a new authentication factor, in addition to the classic three factors: "something the user knows", "something the user has" and "something the user is" which was called the fourth factor: "Somebody the user knows". This factor which is based on social relationships that individuals establish with the people around them is transferred to the realm of computer systems through the vouching process.

This project takes as starting point the fourth factor and the vouching concept to develop an emergency authentication system for the widely known content management system Joomla!

Keywords: authentication, voucher, authentication factors, Joomla

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

"La determinación del objetivo es el punto de partida del logro"

W. Clement Stone

1.1 Introducción

En las últimas décadas, la información ha ido cobrando un papel cada vez más importante en el día a día de las empresas, hasta llegar a convertirse en un activo clave. Una fuga de dicha información, puede llegar a suponer en muchos casos grandes pérdidas para las organizaciones, tanto a nivel económico como en lo relativo a su reputación.

Para que la información sea de utilidad a las empresas, ésta debe ser explotada convenientemente, para lo cual, además de las herramientas adecuadas, se requiere que sea accesible. Es prácticamente imposible encontrar hoy en día una organización que no tenga información en sistemas accesibles a través de la red. Esta tendencia, ha dado lugar a nuevos paradigmas en el mundo de las tecnologías, como por ejemplo la nube o *cloud*, por los que se está apostando fuertemente en los últimos años.

Pero la información no es lo único que se expone en la red. Cada vez son más las compañías que ofrecen servicios de toda índole a consumidores y *partners* a través de Internet. Por ejemplo, hoy en día prácticamente todas las entidades financieras ofrecen servicios de banca electrónica *online* a sus clientes.

A raíz de esta tendencia, surge la necesidad de proteger la valiosa información y los servicios críticos de una forma adecuada. El tradicional binomio formado por usuario y contraseña ya no es suficiente por sí mismo. Este hecho ha motivado la investigación en el ámbito de la autenticación y ha promovido el desarrollo de medidas de seguridad adicionales, con el objetivo de reforzar las ya existentes. Un ejemplo de esto es la aparición de la denominada autenticación multi-factor. Tradicionalmente, los factores de autenticación se resumen en tres: “Algo que el usuario conoce”, “algo que el usuario tiene” y “algo que el usuario es” (también conocido como biometría). La autenticación multi-factor, como su propio nombre indica, hace uso de dos o más factores.

En 2006 un equipo de investigadores de RSA *Laboratories* formado por John Brainard, Ari Juels, Ronald L. Rivest, Michael Szydlo y Moti Yung hablan por primera vez de un cuarto factor: “Alguien a quien el usuario conoce” (1), el cual se basa en el componente social y en el concepto de *vouching*. Por sus características, esta técnica ha sido aplicada como método de autenticación emergencia, sustituyendo a mecanismos más

tradicionales para recuperar el acceso a una cuenta como las preguntas reto o las llamadas al *Help Desk*. Las bases sobre las que se asienta son que un usuario que ha perdido el acceso a su cuenta (protegida mediante un método de autenticación de doble factor) pueda recuperarlo a través de otra persona que le reconoce y posteriormente le facilita una contraseña temporal.

El presente proyecto profundiza en la autenticación basada en las relaciones sociales y toma como punto de partida la idea del cuarto factor y del concepto de *vouching* para desarrollar un sistema de autenticación de emergencia para el gestor de contenidos Joomla!

1.2 Motivación

La principal motivación de este proyecto de fin de carrera nace de la inquietud de probar nuevas vías de autenticación poco exploradas hasta el momento. En concreto la autenticación basada en las relaciones sociales, que da lugar a cuarto factor de autenticación: "Alguien a quien el usuario conoce", que completa a los tres factores clásicos: "Algo que el usuario conoce", "algo que el usuario tiene" y "algo que el usuario es" (o biometría).

Por las características que presenta, este tipo de autenticación se postula como mecanismo de autenticación de emergencia, más que como autenticación primaria. Para la prueba de concepto del mismo se ha seleccionado el gestor de contenidos Joomla!, elección que ha venido motivada por la posibilidad de poder ofrecer una alternativa al mecanismo de autenticación de emergencia presente hoy en día.

El gestor de contenidos Joomla! cuenta con la posibilidad de activar la funcionalidad de autenticación de doble factor, tanto para el *back-end* como para el *front-end* de la aplicación, desde su versión 3.2. Una vez activada, el usuario es instado a introducir nombre de usuario, contraseña y clave de un solo uso u OTP. Para la generación de la clave, Joomla! ofrece la posibilidad de configurar dos proveedores, *Yubikey* o *Google Authenticator* (este último es el que usará en el desarrollo del proyecto). Frente a la indisponibilidad del segundo factor, es decir cuando el usuario no dispone del dispositivo generador de claves OTP, Joomla! propone como mecanismo de emergencia para recuperar

el acceso a su cuenta la impresión de unos códigos de emergencia de un solo uso que deben ser almacenados en un lugar seguro. Sin embrago, son bien conocidas las implicaciones que presenta el almacenamiento físico de claves y contraseñas si no se dispone de los medios adecuados.

1.3 Objetivos

El principal objetivo del presente trabajo es el análisis, el diseño y la implementación de un mecanismo de autenticación de emergencia basado en el cuarto factor de autenticación, "Alguien a quien el usuario conoce" y en el concepto de *vouching*.

El siguiente objetivo, derivado del anterior, consiste en integrar dicho mecanismo de autenticación de emergencia, con el gestor de contenidos Joomla! Para ello deberá estar configurada la autenticación de doble factor utilizando como proveedor de claves OTP *Google Authenticator*.

Por tanto, a modo de resumen, los objetivos que deben ser alcanzados tras la finalización de este proyecto son los siguientes:

- Realización de las fases de análisis, diseño e implementación de un mecanismo de autenticación de emergencia basado en el cuarto factor de autenticación, "Alguien a quien el usuario conoce" y en el concepto de *vouching*.

- Integración del mecanismo de autenticación de emergencia con el gestor de contenidos Joomla!

1.4 Estructura de la memoria

Con el fin de facilitar la lectura del presente documento, a continuación se detalla su estructura y los contenidos de cada capítulo:

- ***Capítulo 1 - Introducción:*** En este capítulo se introduce el objeto del proyecto, la motivación que ha llevado a su realización y se plantean los objetivos que se persiguen con el mismo. Por último se describe la estructura de la memoria.
- ***Capítulo 2 – Estado del Arte:*** En este capítulo se sientan las bases del concepto de autenticación. Asimismo, se introducen conceptos relevantes para la comprensión del proyecto a desarrollar y se analizan las alternativas existentes hoy en día en cuanto a los mecanismos de autenticación de emergencia.
- ***Capítulo 3 – Análisis:*** En este capítulo se analiza la situación actual y se introduce la solución al problema planteado. Posteriormente se estudian los elementos disponibles para abordar los objetivos definidos. Se establece una arquitectura preliminar de lo que será el sistema, se lleva acabo el análisis de requisitos y finalmente se define el plan de pruebas.
- ***Capítulo 4 – Diseño detallado:*** En este capítulo se especifica la arquitectura final del sistema con todos sus componentes. Además se lleva acabo el diseño de clases y los diagramas de secuencia que permitan comprender las distintas interacciones que tienen lugar entre los distintos componentes y clases del sistema.
- ***Capítulo 5 – Implementación:*** En este capítulo se detallan las decisiones más relevantes tomadas durante la fase de implementación, así como los resultados de las pruebas de aceptación definidas en el capítulo de Análisis.
- ***Capítulo 6 – Conclusiones y líneas futuras:*** En este capítulo se presentan las conclusiones obtenidas tras la finalización de este proyecto y se proponen un conjunto de vías que pueden ser analizadas y desarrolladas en futuros proyectos.
- ***Anexo I – Gestión del proyecto:*** En este anexo se detalla la planificación y seguimiento del proyecto. Por otro lado, se incluye el presupuesto junto con las desviaciones detectadas.
- ***Anexo II – Manual de instalación:*** En este anexo se detallan los pasos a seguir para la instalación de la solución desarrollada.

Capítulo 2

Estado del Arte

Una vez definidos la motivación y objetivos principales de este proyecto, en este capítulo se introducen los fundamentos básicos de la autenticación. Primeramente, en la sección 2.1 se ubica la autenticación dentro del contexto del control de accesos y se menciona la relación de ésta con los procesos de identificación y autorización, para proceder a explicar en qué consiste el ciclo de vida del proceso de autenticación en la sección 2.2. En el apartado 2.3 se presentan los tres factores de autenticación clásicos, explicando los pros y contras y los mecanismos de autenticación más extendidos para cada uno de ellos. En el apartado 2.4 se introduce el concepto de autenticación de emergencia y se describen algunos de los mecanismos de recuperación de acceso más comunes a día de hoy en Internet. Por último se exponen las bases sobre las que se asienta este proyecto. En la sección 2.5 se revisa la literatura existente relativa a la autenticación basada en las relaciones sociales y se define el concepto de vouching. Por último, se comentan algunos ejemplos que ya han puesto en práctica este tipo de autenticación.

2.1 La importancia de la autenticación en el control de accesos

La identificación, junto con la autenticación, es un mecanismo que las personas realizamos de forma natural cada día a lo largo de nuestras vidas. Reconocemos a nuestros familiares y conocidos con sólo verlos en persona o mediante una fotografía. También realizamos otros tipos de identificación mediante otros factores como la voz, cuando hablamos con un conocido por teléfono, e incluso podemos identificar quien ha escrito un texto con tan sólo ver la letra o por cómo está escrito. También podemos identificar a otro por su olor, el tacto, su comportamiento, etc. Este proceso de identificar y autenticar a las personas de nuestro entorno, las realiza el cerebro humano con tal sencillez y rapidez que pasan desapercibidas.

La gran expansión que han sufrido las Tecnologías de la Información y los servicios *online* en los últimos años, han trasladado del mundo real al ámbito de los sistemas informáticos la necesidad de identificar, autenticar y autorizar de forma precisa a los usuarios. La identidad del individuo pasa a constituir un elemento valioso y merecedor de la protección adecuada. El hecho de disponer de una identidad propia en la red, permite al usuario hacer uso de los recursos y servicios disponibles, así como garantizar sus derechos a través de este medio. Por tanto, establecer un control de accesos que garantice que el usuario es el propietario legítimo de dicha identidad, se convierte en un elemento crítico.

Hoy en día, el control de accesos es el componente central de cualquier infraestructura de seguridad. Su función es, como su propio nombre indica, controlar los accesos a los recursos y a la información contenida en ellos evitando los accesos no deseados y registrando las acciones realizadas por los usuarios.

Se define acceso como el flujo de información entre un sujeto y un objeto. Un sujeto es una entidad activa (p. ej. un usuario, programa, proceso, etc.) que solicita acceso a un objeto o entidad pasiva o a la información contenida en el mismo (p. ej. un ordenador, base de datos, archivo, programa, etc. (2)

El proceso a través del cual a un sujeto se le otorga acceso a un objeto, se compone de tres pasos fundamentales:

- **Identificación:** es el método mediante el cual el sujeto se identifica ante el sistema. Por ejemplo un nombre de usuario.

- **Autenticación:** es el método mediante el cual se prueba que el sujeto es realmente quién dice ser. Por ejemplo una contraseña o un PIN.

- **Autorización:** es el método mediante el cual se controla a qué objetos puede acceder y qué acciones puede realizar el individuo autenticado sobre dichos objetos. Por ejemplo eliminar un fichero.

Ante esta situación, el proceso natural de identificar y autenticar al usuario pasa a requerir de mecanismos alternativos de autenticación que permitan a los sistemas informáticos testiguar que individuo es quien dice ser.

Como ejemplo para ilustrar este proceso, se plantea la situación en la que un individuo desea sacar dinero de un cajero automático. En primer lugar, el cajero solicita la introducción de la tarjeta bancaria con el fin de identificar al individuo. Previamente a realizar una operación el individuo debe probar que es el propietario legítimo de dicha tarjeta mediante la introducción de un PIN o contraseña que sólo él conoce. Una vez autenticado, de manera implícita, se lleva a cabo una comprobación de que el individuo dispone de los permisos adecuados para retirar el efectivo solicitado. A pesar de que en el contexto de este proyecto no se va a tratar la autorización, es importante reconocer su vinculación con el proceso de autenticación.

2.2 El ciclo de vida de la autenticación

La autenticación no es un evento aislado que se produce en un determinado momento en el tiempo, si no que conlleva un conjunto de actividades y procesos que deben realizarse para el correcto mantenimiento de las soluciones de autenticación y del nivel de seguridad de los sistemas (3):

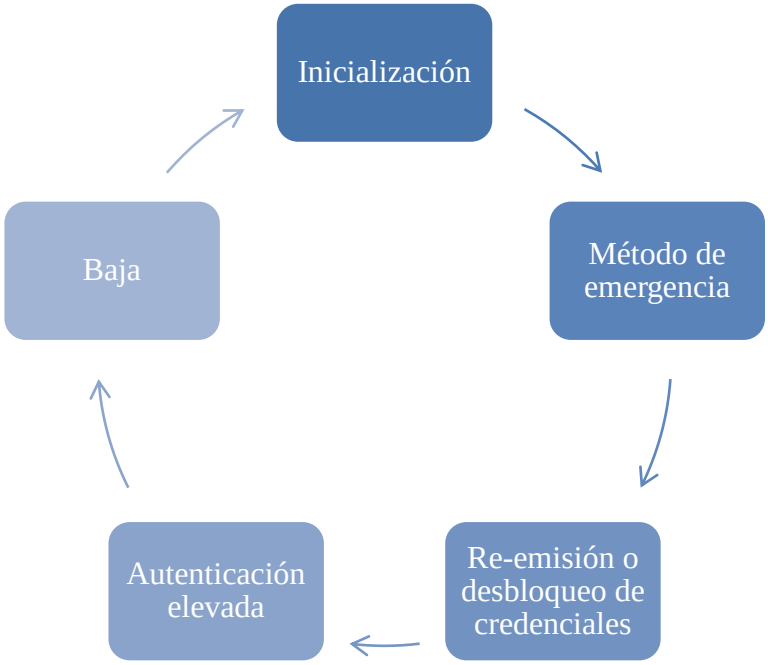

Figura 1: El ciclo de vida de la autenticación (Gartner, 2013)

El proceso de **inicialización** hace referencia al momento en el que se registra al usuario en el sistema y se le conceden las credenciales de acceso al mismo por primera vez. En general, es en este momento cuando también se crean los permisos de acceso de acuerdo a las características del usuario o al rol que va a desempeñar en el sistema.

El **método de emergencia** entra en juego cuando el usuario que necesita acceder a un recurso no dispone del factor de autenticación en ese momento. Cuando por ejemplo un usuario no recuerda su contraseña o no porta consigo su generador de claves, normalmente existe un mecanismo de emergencia que le permitirá acceder al recurso temporalmente de manera segura.

La **re-emisión o desbloqueo de credenciales** se lleva a cabo bien cuando una credencial ha expirado, por ejemplo un certificado electrónico, o ha sido bloqueada para impedir que ésta sea comprometida, por ejemplo tras introducir un número repetido de veces una contraseña incorrecta.

El término **autenticación elevada** aplica a aquellos usuarios que necesitan acceder a un recurso que contiene información especialmente sensible y para el cual es necesario un nivel adicional de autenticación.

La **baja** de un usuario implica la revocación satisfactoria de sus credenciales. Debe realizarse lo antes posible, una vez que el usuario ya no requiera acceder de nuevo al sistema.

2.3 Factores de Autenticación

Los medios que un individuo emplea para autenticarse en un sistema se categorizan en tres grupos según la forma. Cada uno de estos grupos se denomina factor de autenticación.

- **Factor de conocimiento (*Algo que el usuario conoce*):** Se basa en algo que sólo el usuario legítimo sabe. Por ejemplo una combinación usuario y contraseña, un PIN, un patrón de imágenes, etc.

- **Factor de posesión (*Algo que el usuario tiene*):** Se basa en algo que sólo el usuario legítimo tiene. Por ejemplo una tarjeta de identificación, un *token* de generación de claves, un teléfono móvil, etc.

- **Factor de inherencia (*Algo que el usuario es*):** Se basa en alguna característica que es inherente al usuario legítimo o un comportamiento que le diferencia unívocamente del resto. Por ejemplo la huella dactilar, la secuencia del ADN, el iris, la voz, la dinámica del tecleo, etc.

En los últimos años, debido a la expansión de los dispositivos móviles y a la creciente tendencia BYOD (del inglés *Bring Your Own Device*) han surgido nuevos factores, que si bien no tienen la autonomía necesaria para considerarse lo suficientemente seguros por sí mismos, sí que complementan a los factores descritos anteriormente dotándoles de un nivel de seguridad adicional. Un ejemplo de lo anterior es el *fingerprinting* de dispositivos, que en el momento de la autenticación puede detectar si se está realizando desde el dispositivo habitual del usuario (basándose en una huella única generada por la unión de distintos parámetros de la máquina) o por ejemplo la geo-localización del individuo que intenta acceder a un recurso, dado que muchos dispositivos móviles ya llevan incorporado un GPS (4).

En su artículo, titulado "*Fourth Factor Authentication: Somebody You Know*", Ronald Rivest presenta un cuarto factor de autenticación basado en "alguien al que el individuo conoce" (1). Éste factor se basa en una relación de confianza existente entre el individuo que se desea autenticar y la persona que le autentica en el sistema. Se verá en mayor detalle en la sección 2.5 de esta memoria.

Cada factor de autenticación tiene sus fortalezas y sus debilidades, sus partidarios y sus detractores, pero la conveniencia de cada uno (o la combinación de ellos), viene determinada por factores externos como el tipo de sistema que se quiere proteger, el grado de sensibilidad de la información contenida en él o el tipo de usuario entre otros.

2.3.1 Factor de conocimiento

Los mecanismos de autenticación basados en el factor de conocimiento, son aquellos en los que se requiere que el usuario proporcione algo que sólo él, como usuario legítimo, conoce.

Pese a no ser un método tan seguro como otros, actualmente la combinación de usuario y contraseña sigue siendo el método de autenticación más extendido en Internet (5). Esto es así debido en gran medida a la gran aceptación por parte del usuario ante este tipo de autenticación. Es un método sencillo de utilizar y que los usuarios conocen. Los usuarios no tienen que llevar ningún elemento adicional consigo para autenticarse y no resulta intrusivo para el usuario, en el sentido de que no es necesario almacenar ningún tipo de información de carácter personal o física del usuario. Otra característica que hace conveniente su uso, es la sencillez con la que pueden implantarse este tipo de sistemas.

A pesar de su conveniencia, cabe destacar los elevados costes derivados del mantenimiento de las contraseñas, especialmente en las organizaciones donde, a diferencia de Internet, los sistemas self-service para la recuperación de contraseñas todavía no están muy extendidos. Se estima que aproximadamente entre un 10% y un 30% de las llamadas al *Help Desk* están relacionadas con el mantenimiento de contraseñas (6), y que el coste por llamada asciende a entre $25-$30 (7). A la vista de estos números, se pueden inferir los elevados costes a los que tienen que hacer frente las organizaciones en relación a la gestión de las contraseñas de sus empleados.

Los estudios demuestran que una contraseña es más segura cuanto mayor es su complejidad, entendiendo por ésta la combinación de distintos tipos de símbolos (minúsculas, mayúsculas, dígitos y caracteres especiales), que siga un patrón poco predecible (evitar usar el nombre o parte de él, el número de teléfono, el año de nacimiento, etc.) y que por supuesto tenga una longitud mínima (5). Sin embrago, mayor complejidad de la contraseña implica a su vez una mayor probabilidad de que el usuario no sea capaz de recordarla en un futuro. Esto se ve agravado por el hecho de que hoy en día los usuarios deben manejar un elevado número de contraseñas, así como por las restrictivas políticas de seguridad que se establecen en las organizaciones, lo cual lleva a los usuarios a emplear malas prácticas con el fin de recordarlas, facilitando así la labor del atacante (5). Una vez que la contraseña está comprometida, ya no importa la complejidad de la misma. Estudios revelan que de los usuarios que emplean malas prácticas de seguridad, un 25% almacena sus contraseñas en un archivo en el propio ordenador, el 22% lo almacena en otro dispositivo manual, tipo PDA, y un 15% escrita en papel (7).

Además de los ataques de fuerza bruta o diccionario, existe otro tipo de amenaza destinada al robo de contraseñas e información sensible en general que se denomina *Phishing*. Generalmente, mediante el envío de correos electrónicos y mensajes de texto engañosos el usuario es instado a introducir nombres de usuario, contraseñas y otros datos de carácter personal. Éste es un claro ejemplo de la denominada “Ingeniería social” que se aprovecha de aquellos usuarios con escasos conocimientos en materia de seguridad que navegan por Internet. Para tratar de evitar este tipo de ataques, no es suficiente sólo con instaurar avanzadas tecnologías de seguridad, sino que es fundamental que exista un programa de concienciación de los usuarios apropiado (8).

Para tratar de minimizar otro tipo de ataques, tales como el escaneo de los paquetes de red o ataques a los sistemas de almacenamiento, es imprescindible aplicar medidas de seguridad básicas tales como evitar la transmisión y el almacenamiento de las contraseñas en claro, utilizando siempre para ello la función *hash* de la misma.

Existen dos vertientes principales de investigación en cuanto a los mecanismos de autenticación basados en contraseñas. Por un lado se encuentran aquellos estudios enfocados a la habilidad para romper las contraseñas mediante técnicas de ataque, con poco énfasis términos de usabilidad. Por otro, aquellos centrados en la propia usabilidad, es

decir, como generar contraseñas más fáciles de recordar, aumentando así la satisfacción del usuario, pero descuidando aspectos de la seguridad. Existe una vertiente que afirma que usabilidad y seguridad van de la mano, ya que como se ha comentado en párrafos anteriores, si el usuario no es capaz de recordar todas sus contraseñas recurrirá a malas prácticas poniendo en riesgo la seguridad de todo el sistema. Por tanto los estudios concluyen que es imperativo que los desarrolladores se centren por igual en ambos aspectos, usabilidad y seguridad, desde el inicio del ciclo de vida del sistema de autenticación (9).

2.3.2 Factor de posesión

El factor de posesión comprende aquellos mecanismos de autenticación que se sustentan en algo que sólo el usuario tiene. Un ejemplo típico de este factor, en el mundo real, sería una llave que abre una cerradura. El principio básico sobre el que se asienta es que la llave guarda un secreto que sólo comparte con la cerradura. Este mismo principio es aplicado a los sistemas informáticos. La seguridad reside tanto en la integridad de la parte autenticadora (la cerradura), como en la protección física del factor de posesión (la llave).

Ejemplos comunes de mecanismos de autenticación de este tipo son las claves de acceso basadas en tiempo u OTP (del inglés *One Time Password*). Estas contraseñas perecederas de un solo uso, también denominadas comúnmente *tokens*, son generadas por un dispositivo hardware que está sincronizado con el sistema de validación y utiliza la misma semilla aleatoria que éste: el tiempo. Es un mecanismo que se ha hecho muy popular ya que ha demostrado ser una solución muy eficiente para solventar el problema de la usurpación de identidad. En los últimos años han surgido diversas variantes de los generadores de claves OTP, en forma de tarjeta de crédito, con teclado incorporado, con lectores de huellas digitales e incluso soluciones software que los emulan de forma virtual. Estos últimos han demostrado altamente eficientes, ya que al poder ser instalados en dispositivos móviles o en el propio ordenador, mejoran aspectos como el coste asociado al hardware o la satisfacción del usuario al no tener que portar elementos adicionales. El mecanismo de autenticación comercial basado en claves OTP más conocido es *SecurID* de RSA, disponible tanto en versión *hardware* como *software*. *Google Authenticator* es una alternativa gratuita para dispositivos móviles ampliamente extendida.

Otros mecanismos de autenticación bajo esta categoría son las tarjetas inteligentes o *Smart Cards*, tarjetas de plástico con un circuito integrado embebido. Son capaces de autenticar, almacenar datos y procesar aplicaciones. Su uso es más común en grandes organizaciones. Los certificados digitales X.509, normalmente contenidos en *Smart Cards* o dispositivos USB, son otro método de autenticación ampliamente extendido tanto en las organizaciones como en Internet. Este método hace uso de una infraestructura de clave pública (PKI) para autenticar al usuario. Una vez que el usuario recibe su certificado, emitido por una autoridad de certificación (CA) (la cual puede ser propia o un proveedor externo), puede utilizarlo para acceder de manera segura al sistema. La autenticación mediante certificados es un método robusto y utilizado a desde hace años. Sin embargo, su uso no es compatible con los dispositivos móviles, debido a que carecen de lectores de tarjetas o puertos USB.

Aunque los mecanismos de autenticación basados en el factor de posesión se consideran suficientemente seguros, existen múltiples formas de ataque. Al tratarse de un elemento físico, el atacante sencillamente puede robar el factor de posesión. Por ejemplo un *token* generador de claves puede ser robado y utilizado antes de que el propio usuario sea consciente del robo. El segundo factor no es muy efectivo frente a ataques *Man in the Middle* (MITM), en los que el atacante podría interceptar la comunicación hacerse pasar por la parte autenticadora, es por ello que normalmente el segundo factor no se utiliza como factor de autenticación aislado, sino en conjunto con otro. Es posible conocer el secreto compartido con la parte autenticadora atacando por ejemplo la base datos en la que se almacena y aplicando ingeniería inversa o bien interceptándolo durante la comunicación. Por otro lado, el atacante podría realizar una copia del factor de posesión si éste no se protege de manera adecuada y obtener un duplicado del mismo. Sin embargo, hoy en día, la gran mayoría de las soluciones comerciales (*tokens* OTP, *Smart Cards*, etc.) están protegidas contra copia.

A la hora de elegir una solución de autenticación basada en el factor de posesión se deben tener en cuenta una serie de implicaciones con el fin de evaluar si es la solución más adecuada: en primer lugar destacar los costes asociados a los dispositivos hardware, que suelen ser altos. Requiere labores de mantenimiento y distribución, por lo que influye enormemente el número y tipo de usuarios que van a utilizar el sistema. La aceptación por parte del usuario final es otro aspecto a tener en cuenta. Hoy en día las personas

transportamos numerosos elementos con nosotros como tarjetas de crédito, tarjeta de acceso al lugar de trabajo, las llaves de casa o del coche, dispositivos móviles, etc. Tener que llevar consigo un elemento más representa para muchos usuarios un inconveniente (10). Sin embargo, la tendencia a utilizar el teléfono móvil, algo que el usuario ya lleva consigo de por sí, como factor de posesión para la autenticación, no sólo ha ayudado a mitigar la pobre aceptación por parte de los usuarios hacia este tipo de métodos, sino que también ha reducido considerablemente los costes para las organizaciones (10).

2.3.3 Factor de inherencia o biométrico

El tercer factor de autenticación hace referencia a lo que el usuario es, es decir, el usuario se autentica mediante alguna característica inherente a él que lo diferencia unívocamente del resto. Este concepto se denomina biometría.

Los mecanismos de autenticación biométrica se dividen en dos tipos. Por un lado los basados en características fisiológicas del individuo, como por ejemplo la huella dactilar, el iris, la palma de mano o el rostro, entre otros. Por otro lado están los basados en patrones de comportamiento como el reconocimiento de voz, la dinámica del tecleo o el reconocimiento de la firma manuscrita.

Los mecanismos de autenticación biométrica se componen de un dispositivo hardware que captura una serie de medidas y posteriormente las compara contra un patrón que se encuentra almacenado. Normalmente, estas medidas nunca encajan de manera exacta, por lo que se establece un umbral que invalida aquellas que estén fuera del mismo. Esto da lugar a un nuevo problema que no se producía con los anteriores factores: los falsos positivos y los falsos negativos. Los falsos positivos representan un importante problema de seguridad, mientras que un elevado número de falsos negativos dan lugar a un sistema lento y poco eficiente. La clave de los sistemas biométricos está, por tanto, en encontrar el equilibro entre ambas situaciones (11).

La biometría soluciona algunos problemas relacionados con los otros dos factores, como por ejemplo evita al usuario tener que portar dispositivos adicionales para la autenticación, con los problemas de robo y pérdida que ello conlleva. También termina con el problema de que los usuarios apunten o compartan deliberadamente sus contraseñas, ya que las características físicas no se pueden compartir.

Pese a ser considerado un mecanismo muy seguro, la biometría todavía presenta algunos inconvenientes. Algunos mecanismos de autenticación biométrica dan lugar a soluciones poco escalables por diversas razones. Por un lado, la frecuencia de los falsos positivos tiende a incrementarse según aumenta el número de usuarios, debido a que aumenta la probabilidad de encontrar individuos con características biométricas tan similares que no puedan ser detectadas por el sistema. Por otro lado, en muchos casos se requiere almacenar cantidades elevadas de información por usuario registrado, lo cual en entornos con un número muy elevado de usuarios podría llegar a disparar los costes de almacenamiento. Asimismo, ciertos tipos de sensores biométricos representan un coste inaceptable para muchas empresas. Un ejemplo de este tipo de sensores son los oculares, utilizados en el reconocimiento de iris o de retina (11).

Existen usuarios que pueden considerar intrusivo el hecho de que sus características físicas queden registradas en un sistema y puedan llegar a ser utilizadas sin su conocimiento. Un típico ejemplo es el de la firma manuscrita digitalizada, que aunque hoy en día está presente en un gran número de actividades diarias que realizamos, en sus inicios produjo el rechazo de muchos usuarios. La forma en la que se captura la información biométrica también puede provocar el rechazo de los usuarios, bien por razones culturales o por ser una técnica incómoda por el usuario, por ejemplo el reconocimiento del ADN o de la retina.

Algunos tipos de huella biométrica pueden ser alterados por factores externos. Por ejemplo, el reconocimiento de la huella dactilar puede verse afectado por accidentes tales como quemaduras o la presencia de lociones, grasa, etc. El reconocimiento de la voz también puede alterarse debido a un resfriado u otras enfermedades, el estrés, etc.

Hasta hace poco, la biometría se restringía a entornos de alta seguridad, como organismos gubernamentales, hospitales o el ámbito militar. Sin embargo con la introducción de los lectores de huella dactilar en los dispositivos móviles, la tendencia es utilizar esta tecnología cada vez más en las transacciones diarias de los usuarios tales como comercio electrónico, banca electrónica y demás servicios disponibles a través de la red. Sin embargo, al mismo tiempo que ganan popularidad, los atacantes obtienen más conocimientos acerca del funcionamiento de los distintos métodos de autenticación biométrica y creando técnicas de ataque más sofisticadas contra estos sistemas. Por

ejemplo, existen técnicas, aunque muy costosas, que permiten clonar una huella dactilar impresa en una superficie, por ejemplo mediante la toma de una fotografía con una alta resolución.

Uno de los grandes interrogantes que arroja la biometría, es qué hacer una vez que la información biométrica ha sido comprometida, ya que ésta es muy difícil de cambiar. Una contraseña que se ve comprometida puede ser fácilmente sustituida por otra, sin embargo la huella dactilar es una característica inherente al usuario y no puede ser reemplazada.

2.3.4 Autenticación multi-factor

La proliferación del acceso remoto a través de las redes y los dispositivos móviles a datos sensibles y aplicaciones corporativas; el incremento en el uso de *cloud computing*; y el auge del comercio electrónico y la banca *online* están actualmente en el punto de mira de los atacantes. Llegados a este punto, la autenticación de una sola capa, basada tradicionalmente en usuario y contraseña, ya no aporta el nivel de seguridad requerido.

En los últimos años se ha impuesto con fuerza la autenticación multi-factor, especialmente en aquellos sectores que ofrecen servicios críticos a través del canal *online*. La autenticación multi-factor, como su propio nombre indica, impone la necesidad de autenticarse mediante al menos dos factores distintos.

Figura 2. Combinación de los factores de autenticación clásicos

La combinación más extendida en la autenticación de dos factores o de doble factor es un método del primer factor, como es usuario y contraseña, junto con un método perteneciente al segundo factor, normalmente un generador de claves OTP. El Instituto Nacional de Estándares y Tecnología norteamericano (NIST) ha definido una guía para la implementación de mecanismos de autenticación electrónica remota que define los requisitos a cumplir para logar cada uno de los cuatro niveles de seguridad establecidos. Los mecanismos de autenticación de dos factores logran un nivel de seguridad 3, definido como el nivel de seguridad apropiado para aquellas transacciones que requieren un alto grado de exactitud en la verificación de la identidad (12).

La autenticación de tres factores proporciona el mayor grado de seguridad posible. Consiste en añadir un método de autenticación biométrico a los dos anteriores. Este tipo de autenticación normalmente se reserva para entornos de alta seguridad en los que se maneja información extremadamente sensible, es por ello que su uso está más extendido en los ámbitos gubernamental, militar y sanitario.

2.4 **Métodos de autenticación de emergencia**

Cabe esperar que en determinadas ocasiones, el usuario que quiere acceder a su cuenta no disponga del método de autenticación primario. Por ejemplo, es común que un usuario olvide cuál es su contraseña de acceso o no porte consigo su dispositivo móvil o su *token* generador de claves. En estos casos deben establecerse mecanismos de autenticación alternativos que permitan al usuario recuperar el acceso a su cuenta.

En general todo sistema de autenticación, para ser completo, debe proporcionar al menos un método de autenticación secundario o de emergencia. Para que la seguridad global del sistema no se vea afectada, el método de autenticación de emergencia debe ser al menos tan seguro como el método primario. Un fallo de autenticación en el método de emergencia puede acarrear importantes consecuencias. En el caso de denegar el acceso al usuario legítimo, éste podría perder el acceso a su cuenta de forma permanente. Por otro lado, si método de emergencia es débil y es propenso a otorgar acceso a un usuario no

legítimo, puede ser utilizado como blanco por los atacantes, viéndose comprometida la seguridad del sistema.

El éxito de todo método de autenticación viene determinado por las siguientes características (13):

- **Costes de configuración y mantenimiento:** el tiempo y el esfuerzo empleados por el usuario para configurar o reconfigurar el mecanismo de autenticación.

- **Eficiencia:** el tiempo y el esfuerzo empleados por el usuario cada vez que se autentica en el sistema.

- **Fiabilidad:** la probabilidad con la que el usuario puede probar su identidad en el sistema.

- **Seguridad:** el tiempo y el esfuerzo requeridos para suplantar con éxito la identidad del usuario legítimo o bien la probabilidad de conseguirlo.

No hay un mecanismo de autenticación perfecto que cumpla a rajatabla las características anteriores. Sin embargo, para que el mecanismo tenga éxito debe existir un grado de cumplimiento mínimo de todas ellas. En función del propósito al que vaya destinado el método de autenticación, se dará mayor relevancia a unas características que a otras.

Dadas las consecuencias ante un fallo en el mecanismo de autenticación de emergencia, detalladas en párrafos anteriores, la fiabilidad y la seguridad son aspectos fundamentales a tener en cuenta en el diseño de un sistema de este tipo. Dado que la autenticación de emergencia es un mecanismo que se emplea sólo en determinadas ocasiones, a diferencia del mecanismo de autenticación primario, la eficiencia es una característica menos relevante.

En general, los sistemas de autenticación de emergencia se pueden dividir en dos categorías: aquellos basados en algo que el usuario conoce, en los cuales el usuario debe proporcionar cierta información al servidor en el momento de autenticarse y autenticación transitiva, en los cuales el sistema delega la autenticación en otro sistema diferente (14).

2.4.1 Autenticación de emergencia basada en algo que el usuario conoce

Los mecanismos de autenticación de emergencia basados en algo que el usuario conoce están muy extendidos debido a que son sencillos de implementar y no requieren de ningún tipo de infraestructura adicional. A continuación se hace un breve repaso de los más comúnmente utilizados:

Preguntas de seguridad o preguntas reto

Las preguntas reto es un claro ejemplo de este tipo de mecanismos de autenticación de emergencia. La clave de este método reside en que las preguntas deben ser fáciles de responder para el usuario, pero al mismo tiempo las respuestas deben ser difíciles de adivinar para el atacante.

A pesar de que su uso está ampliamente extendido en Internet, este método presenta algunas deficiencias de seguridad: en los casos en que las preguntas no son configurables, pueden no aplicar al usuario. Como por ejemplo la pregunta: "¿Cuál es el nombre de tu mascota?, ya que no todos los usuarios tienen que tener o haber tenido una mascota. Existen ciertas preguntas cuya respuesta puede variar con el tiempo, como por ejemplo las relacionadas con las preferencias. Ante la pregunta "¿Cuál es tu canción favorita?", el usuario puede llegar a olvidar cual era la respuesta que dio en su momento ya que puede darse el caso de que sus preferencias hayan cambiado. Otras respuestas simplemente pueden ser fácilmente adivinables por el atacante. Por ejemplo, ante la pregunta "¿Cuál es tu personaje histórico preferido?", cabría esperar que en Estados Unidos una gran mayoría de las personas respondieran que es "George Washington".

Además de los ejemplos citados en el párrafo anterior, con la aparición de las redes sociales, las preguntas reto se han convertido en un mecanismo de autenticación aún más débil. Muchas preguntas pueden ser fácilmente rastreables as través de los perfiles de los usuarios, en los que cada vez hay disponible más información de carácter personal.

Por lo tanto, los esfuerzos a la hora de diseñar un mecanismo de autenticación basado en preguntas reto, deben ir dirigidos a que las respuestas a las preguntas sean fáciles de recordar, no rastreables en la red, no sean populares entre distintos usuarios y que sea

desconocida para contactos del usuario no confiables (14). Algo que, a la vista de los estudios publicados, reduce considerablemente el abanico de preguntas disponible.

Secreto compartido impreso

Debido al hecho de tener que manejar un gran número de contraseñas distintas y a la complejidad de las mismas, es frecuente que muchos usuarios las escriban en papel con el fin de poder recordarlas en un futuro. Lejos de condenar esta conducta, la cual presenta ciertas implicaciones de seguridad pero es comprensible por otro lado, las investigaciones hacia nuevos métodos de autenticación han explorado esta vía, creando sistemas diseñados expresamente para manejar claves impresas de forma segura.

Para ello, el sistema genera una matriz de claves impresas y requiere en el momento de la autenticación un subconjunto aleatorio de estas claves, que pueden estar identificadas por un índice o coordenada. Las ventajas de este mecanismo frente a las preguntas reto, es que se elimina la posibilidad de que el atacante adivine la respuesta a través de prácticas tales como rastrear en la red o análisis estadísticos que den con las respuestas más populares ante una determinada pregunta. También se reducen los ataques de tipo MITM (del inglés *Man In The Middle*).

Contraseñas anteriores

Normalmente existen políticas de seguridad que obligan a cambiar las contraseñas pasado un determinado período de tiempo. Las contraseñas anteriores pueden utilizarse en conjunción con otros mecanismos de autenticación de emergencia para recuperar el acceso a una cuenta. Por ejemplo, una vez comprometidas las preguntas reto por un atacante puede solicitarse la contraseña anterior de la cuenta, la será conocida únicamente por el usuario legítimo, nunca por el atacante que ha comprometido la cuenta.

2.4.2 Autenticación de emergencia basada en métodos transitivos

Los mecanismos que recaen bajo esta clasificación reciben su nombre debido a traspasan el testigo de autenticar al usuario a otros sistemas.

Autenticación mediante correo electrónico

El correo electrónico es el mecanismo de autenticación transitiva más común en Internet. La clave reside en el supuesto de que sólo el usuario legítimo puede acceder a un secreto enviado a su cuenta de correo. Normalmente, esta clave secreta va embebida en una URL dentro del correo de tal forma que el usuario no tenga que introducirla manualmente.

Trasladar la autenticación al proveedor de correo electrónico tiene sentido dado que normalmente la autenticación en los servidores de correo es lo suficientemente segura, ya que el correo alberga información en muchos casos de carácter personal o sensible. Sin embargo, este método presenta ciertas limitaciones tales como que el usuario introduzca mal su dirección de correo o que la dirección de correo proporcionada ya no esté en uso o lo que es aún peor, haya sido asignada a otra persona.

Autenticación mediante el teléfono y otros dispositivos móviles

Al igual que sucede con las direcciones de correo electrónico, los números de teléfono pueden ser utilizados para este tipo de autenticación. Los sitios web pueden enviar mensajes de texto o utilizar sistemas automatizados de voz para llamar al usuario y proporcionarle el código de acceso. En este caso, la seguridad y fiabilidad del sistema recae en la propia seguridad del teléfono móvil, la cual es más débil que la seguridad implementada por un servidor de correo. El riesgo de perder el teléfono hace que la seguridad proporcionada por este mecanismo sea en muchos casos insuficiente.

2.5 Autenticación basada en las relaciones sociales

El empleo de las relaciones interpersonales para efectuar la identificación y autenticación de personas no es un concepto nuevo. El proceso de presentar una persona a otra es la forma más común de identificar (e implícitamente autenticar) al otro. Sin embargo, en el ámbito de la seguridad informática, el mecanismo natural de identificación mediante las relaciones sociales ha sido poco explorado formalmente en términos de autenticación.

La reducida literatura existente propone el proceso de *vouching* como forma de implementar la autenticación basada en las relaciones sociales. El ***vouching*** es el proceso mediante el cual un usuario hace uso de su identidad para ayudar a otro usuario a acceder a un sistema. A partir de ahora se denominará **ayudante** al usuario que asiste y **solicitante** al usuario que requiere acceder al sistema. Por sus características, este método es adecuado para ser usado como mecanismo de autenticación de emergencia.

En el *paper* titulado *"FourthFactor Authentication: Somebody You Know"* escrito por J. Brainard *et al.*, se describe un prototipo de un sistema de *vouching* para ser utilizado como mecanismo de autenticación de emergencia para la conocida solución comercial de autenticación mediante claves OTP, *RSA SecurID* (1).

El prototipo estipula que el proceso es iniciado por el usuario solicitante, el cual establece comunicación con otro usuario asignado como su ayudante mediante un canal *Out of Band* como puede ser el teléfono o en persona. La premisa para asignar a un usuario como ayudante de otro es que ambos usuarios deben conocerse, pudiendo de esta manera el ayudante identificar al solicitante. Una vez que el ayudante ha reconocido al solicitante, éste accederá al sistema de generación de códigos, autenticándose mediante un mecanismo de doble factor para obtener el código *voucher* que será utilizado por el solicitante para recuperar el acceso al sistema objetivo. El código *voucher* nunca deberá ser enviado por correo electrónico o mensaje de texto. Por razones de seguridad debe ser entregado en mano al usuario solicitante o bien comunicado oralmente a través del teléfono (1).

El proceso de vouching descrito en el párrafo anterior tiene cabida en el ámbito organizacional como método de autenticación de emergencia, liberando a los equipos de soporte técnico y *Help Desk* de dicha labor. Sin embargo, a la hora de diseñar un mecanismo de autenticación de este tipo, surgen una serie de cuestiones que no aplican a los mecanismos de autenticación clásicos. Consideraciones como la usabilidad o la ingeniería social son factores importantes a tener en cuenta. Si el sistema es se presenta complicado de utilizar para el usuario es de esperar que éste termine contactando con el *Help Desk*. Por otro lado, es fundamental para la seguridad del sistema que el usuario ayudante identifique correctamente de manera unívoca al usuario solicitante. Si el usuario ayudante recibe solicitudes a través del correo electrónico o llamadas de desconocidos, garantizar sería imposible garantizar la seguridad. Por ello, el *vouching* debe apoyarse en

relaciones de confianza reales en una red de contactos acotada y haciendo uso siempre que sea posible del cara a cara.

Se explicará este punto con mayor detalle en el capítulo de análisis pues es el eje central de este proyecto.

Por otro lado, *Microsoft Research* llevó a cabo un estudio para evaluar la fiabilidad, eficiencia y seguridad de un mecanismo para el restablecimiento de la contraseña basado en las relaciones sociales en su página *Windows Live* (13).

El proceso de configuración del mecanismo de autenticación propuesto por Microsoft requiere que el usuario propietario de la cuenta proporcione el nombre y dirección de correo electrónico de cuatro contactos de confianza. Con el fin de poder recuperar el acceso a su cuenta, el usuario debe obtener un código *voucher* de cada uno de sus contactos de confianza. Al igual que en caso anterior, es recomendable que la comunicación con los contactos de confianza se lleve a cabo por vía telefónica o en persona, nunca por correo electrónico o mensaje de texto, ya que ambos medios son fácilmente falsificables. Cuando el contacto de confianza se conecta a la página web a través de cual obtendrá el código *voucher*, en una primera instancia se le solicitará la razón por la que está accediendo y su consentimiento acerca de que conoce las consecuencias de otorgar un código de acceso a alguien que no sea el propietario legítimo de la cuenta. Una vez aceptado se presentará el código voucher por pantalla y se le insta a que contacte con el propietario de la cuenta por vía telefónica o bien cara a car para proporcionarle el código. Una vez que el usuario tiene los cuatro códigos ya está en disposición de recuperar el acceso a su cuenta (13).

Este mecanismo, aunque también se basa en las relaciones sociales y códigos *voucher* para proporcionar autenticación de emergencia, presenta algunas diferencias con respecto al anterior que hacen más apropiado su uso en sitios web que en organizaciones. Sin ir más lejos, la red social *Facebook* utiliza desde mayo de 2013 este mecanismo de emergencia en su sitio web. El usuario debe seleccionar entre tres y cinco amigos de *Facebook* con los que pueda ponerse en contacto si alguna vez necesita ayuda para acceder a su cuenta (por ejemplo, si olvida su contraseña y además no puede acceder a su cuenta de correo electrónico para restablecerla). *Facebook* aprovecha de esta manera las relaciones

de amistad en la red social para validar que el usuario conoce a sus contactos de confianza y *viceversa*. Es preciso remarcar que para que este mecanismo sea efectivo, además de que el contacto sea una persona de confianza para el usuario, se deben tener en cuenta otros criterios para su selección:

- Debe ser un contacto con el que el usuario pueda contactar fácilmente por teléfono.

- El usuario debe conocer cómo acceder a su cuenta de Facebook. Para ello es preferible que sea un usuario que acceda frecuentemente a su cuenta (de otra forma podría olvidar más fácilmente su contraseña).

- Debe ser un contacto que, dentro de lo que cabe, tenga acceso a la red y pueda entrar en su cuenta de Facebook.

Por último, comentar la existencia de otras vías de investigación que tratan de implementar la autenticación basada en las relaciones sociales y en el concepto de *vouching* mediante el uso de teléfonos y otros dispositivos móviles (15).

Capítulo 3

Análisis

En este capítulo se analiza la problemática que se pretende resolver, es decir, la implementación de un mecanismo de autenticación de emergencia para Joomla! basado en el cuarto factor y en el concepto de *vouching* introducido en el capítulo anterior. Por tanto, en la sección 3.1 se explica en detalle cual es dicho problema para a continuación, en la sección 3.2, pasar a definir la solución propuesta. En la siguiente sección, 3.3, se presenta la arquitectura de componentes a alto nivel que servirá como punto de partida para el resto del análisis. El siguiente paso consiste en realizar un análisis de las tecnologías impuestas y las tecnologías web disponibles, justificando la elección de la tecnología no impuesta que se utilizará en la implementación de la solución. Éste se lleva a cabo en la sección 3.4. Las secciones 3.5 y 3.6 se corresponden con la definición de los casos de uso y de los requisitos de software respectivamente. Por último, en la sección 3.7 se detallan los casos de prueba que determinarán la correcta implementación de los requisitos definidos en las secciones anteriores.

3.1 Planteamiento del problema

El 6 de Noviembre de 2013 el equipo de desarrolladores de Joomla! lanza la versión 3.2.0 de este conocido gestor de contenidos. Una de las funcionalidades más importantes incluidas en esta nueva versión es la posibilidad de activar la autenticación de doble factor, tanto para el *front-end* como para el *back-end* de Joomla! (16).

Impulsado por la creciente demanda del mercado de proteger adecuadamente la información y los servicios expuestos en Internet, la tendencia hacia la utilización de mecanismos de autenticación de doble factor se vuelve casi imprescindible en ciertos sectores que requieren un extra de seguridad. Bien es cierto que hasta el momento existían multitud de extensiones que dotaban a Joomla! de una capa de seguridad adicional proporcionada por el segundo factor de autenticación, pero no es hasta la versión 3.2.0 cuando es implementado de forma nativa.

Cuando se activa el mecanismo de autenticación de doble factor de Joomla! el usuario debe introducir su usuario, contraseña (factor de conocimiento) y una clave OTP (factor de posesión). Joomla! ofrece la posibilidad de configurar la autenticación de doble factor con dos proveedores de claves OTP diferentes *Google Authenticator* o *Yubikey*.

El mecanismo de autenticación de emergencia, frente a la indisponibilidad del factor de posesión, es el mismo tanto en el caso de haber configurado la autenticación de doble factor con *Google Authenticator* como con *Yubikey*. En el momento en el que se activa la autenticación de doble factor, Joomla! solicita la impresión 10 claves de emergencia de un solo uso y recomienda que se guarden en un lugar seguro.

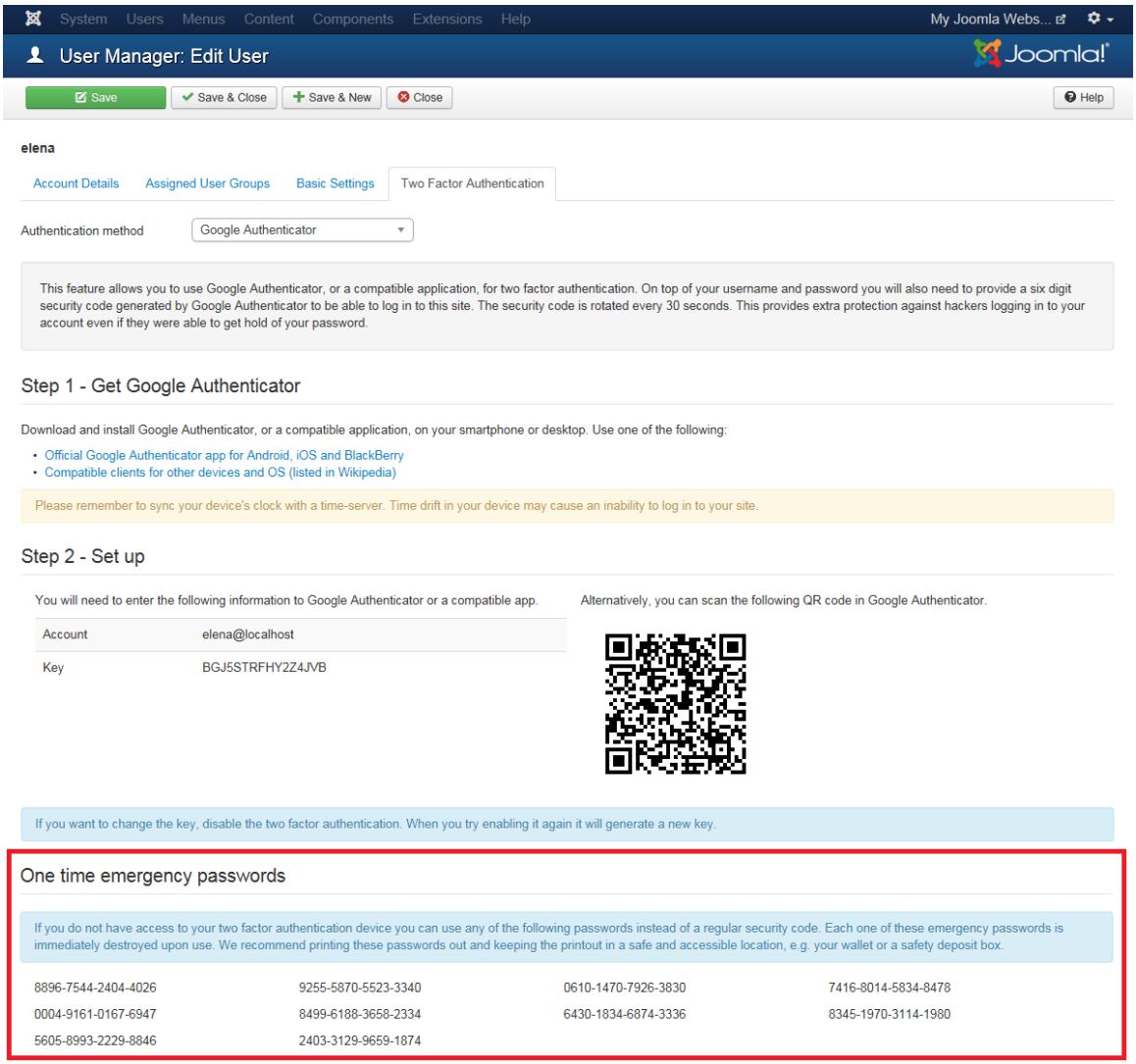

Figura 3. Pantalla de activación de la autenticación de dos factores de Joomla!

El almacenamiento de las contraseñas en un medio físico entraña numerosos riesgos. Si el atacante consigue acceder al lugar en el cual se almacenan las claves de emergencia, eso le otorgaría acceso inmediato al sistema. Asimismo, no existe un mecanismo para restituir las claves de emergencia en caso de que éstas sean comprometidas, lo cual plantea el problema de qué hacer una vez que se han utilizado todas las claves.

Partiendo del hecho de que un sistema de autenticación es tan seguro como el más débil de sus componentes (1), el mecanismo de emergencia debe ser al menos tan seguro como el mecanismo de autenticación primario. Las debilidades presentes en este

mecanismo de autenticación de emergencia impactan directamente en el nivel de seguridad proporcionado por el sistema nativo de autenticación de doble factor de Joomla! convirtiéndolo en un método de autenticación considerado débil en los distintos ámbitos, en los que cada vez cobra más valor la información y se vuelve más crítica la correcta protección de la misma.

3.2 Perspectiva general de la solución

Como solución al problema planteado, se propone un sistema de autenticación de emergencia basado en el cuarto factor de autenticación "Alguien a quien el usuario conoce" y en el concepto de *vouching* introducido en el capítulo anterior. A partir de ahora **solicitante** hace referencia al usuario que ha perdido el acceso a su cuenta en el portal de Joomla! y **ayudante** al usuario que le facilita de nuevo el acceso.

Previamente a iniciar el proceso de *vouching*, es necesario llevar a cabo un proceso de inicialización (o *enrollment*) para cada usuario, en el cual se le otorgan los permisos necesarios para poder acceder al servidor de códigos *voucher* y se establece qué usuario puede actuar como ayudante suyo llegado el momento. En la solución aquí planteada, el proceso de inicialización de los usuarios es llevado a cabo por el rol de administrador del sistema.

El proceso de *vouching* es desencadenado por el usuario solicitante, el cual desea acceder a su cuenta de usuario en el portal Joomla! pero no dispone del segundo factor, en este caso del dispositivo generador de claves *Google Authenticator*. Sin embargo sí recuerda su usuario y contraseña. A continuación se describe la secuencia de acciones que debe llevar cabo para recuperar el acceso al sistema:

1. El solicitante, que conoce quién es su ayudante, contacta él a través de un canal independiente (*Out Of Band*), como puede ser el teléfono o el cara a cara. El correo electrónico queda descartado en este escenario debido a los problemas de seguridad que presenta.

2. El ayudante autentica al solicitante, es decir verifica su identidad, bien reconociendo su voz a través del teléfono o bien reconociéndole físicamente si se encuentran cara a cara.

3. El ayudante se autentica en el sistema de vouching, usando autenticación de doble factor. En este caso usuario/contraseña y clave OTP proporcionada por *Google Authenticator*.

4. Una vez autenticado le debe indicar al sistema la identidad del usuario solicitante al que está ayudando. El sistema debe verificar que la relación ayudante-solicitante es una relación existente y válida. En caso afirmativo, el sistema genera un código *voucher* y registra una sesión de *vouching* activa asociada a ambos usuarios, ayudante y solicitante.

5. El ayudante proporciona al solicitante de forma oral el código voucher generado.

6. El solicitante accede al portal Joomla! e introduce en el formulario de inicio de sesión su usuario, contraseña y el código *voucher* que le ha proporcionado el ayudante. Si existe una sesión de *vouching* válida para dicho solicitante, es decir se cumple que la sesión está activa, no se ha sobrepasado el tiempo de vida de la sesión y el código voucher no ha sido utilizado con anterioridad, el usuario es autenticado satisfactoriamente en el portal.

En la Figura 4 se representa gráficamente el proceso de vouching descrito en los puntos anteriores:

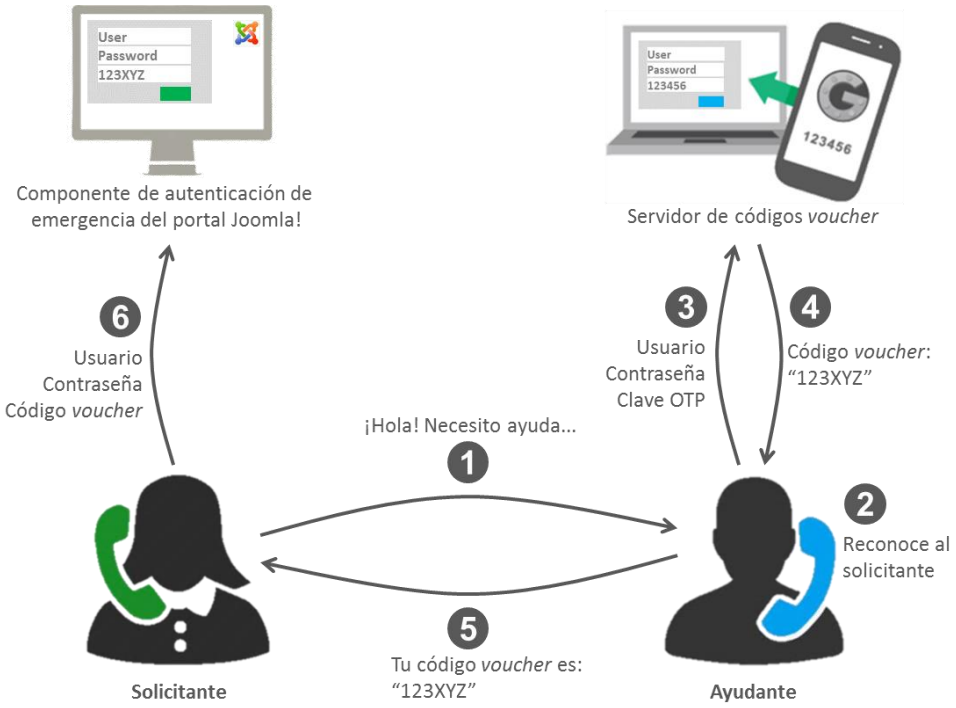

Figura 4. Flujo del proceso de vouching

A la vista del proceso descrito, el sistema a desarrollar debe estar formado por dos partes fundamentales. Por un lado, una aplicación gestora de códigos voucher, a la que a partir de ahora se hará referencia como **servidor de códigos voucher**. Ésta será la aplicación a la que el ayudante deberá acceder para la generación del código *voucher*. Además desde esta aplicación se permitirá llevar a cabo la gestión de usuarios, relaciones ayudante-solicitante y sesiones de *vouching*. Una **sesión de *vouching*** es un registro que se crea en el mismo momento de la generación del código *voucher*, y cuya función es mantener el estado e información asociadas al mismo, como por ejemplo el usuario ayudante que lo ha generado, el solicitante, el estado, la fecha de creación y la de expiración, entre otros. Por otro lado, es necesaria una extensión de Joomla! que permita a un usuario solicitante autenticarse en el portal de acuerdo a lo descrito, es decir mediante su usuario, contraseña y el código voucher proporcionado por el ayudante. A esta extensión se hará referencia como **componente de autenticación de emergencia** del portal Joomla!

Al tratarse de una aplicación a través de la cual los usuarios son capaces de recuperar el acceso a sus cuentas apoyándose en la capacidad del ayudante para poder acceder al sistema, el servidor de códigos voucher debe ser implementado en una tecnología que pueda ser accedida fácilmente por el ayudante. Las aplicaciones web presentan la gran ventaja de que pueden ser accedidas desde prácticamente cualquier lugar y cualquier dispositivo, siempre que se disponga de conexión a Internet. Asimismo, pueden ser utilizadas por varios usuarios al mismo tiempo.

En siguiente figura se presenta la arquitectura del sistema:

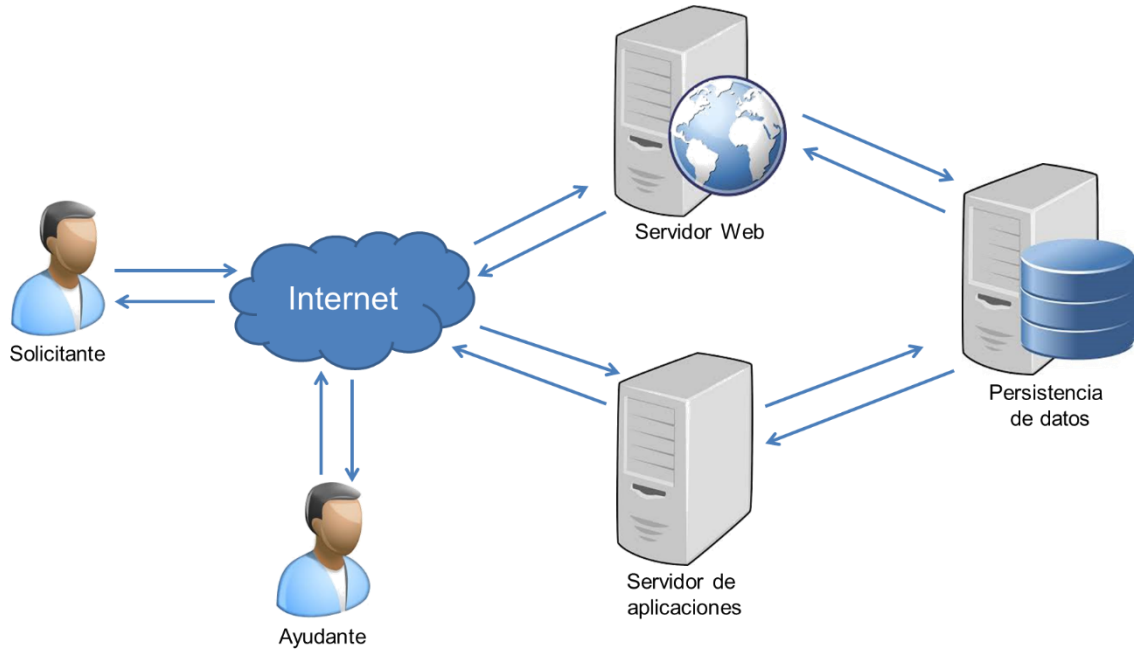

Figura 5. Arquitectura del sistema

La arquitectura propuesta está comprendida por un servidor web donde estará a alojada la instalación de Joomla! junto con el componente de autenticación de emergencia y un servidor de aplicaciones que almacena el servidor de códigos *voucher*. La capa de persistencia del sistema está formada por una base de datos, a la cual acceden tanto el servidor de códigos *voucher* como la extensión de autenticación de emergencia del portal de Joomla!

A través de sus propios dispositivos conectados a internet, el usuario solicitante y el ayudante son capaces de acceder al portal de Joomla! y al servidor de códigos *voucher* respectivamente.

3.3 Arquitectura preliminar

Una vez presentada la solución a desarrollar, se propone la arquitectura preliminar de los componentes que forman el sistema con el objetivo de servir como punto de partida para el posterior diseño.

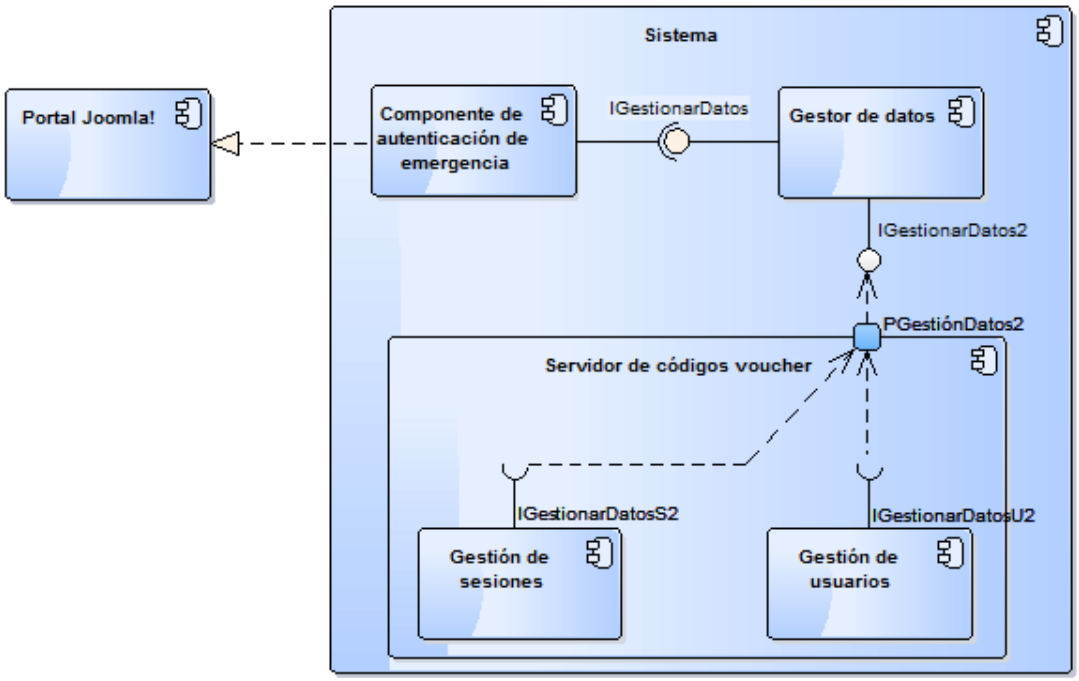

Figura 6. Arquitectura de componentes preliminar del sistema

El la figura anterior se expone el diagrama de componentes que representa la arquitectura preliminar del sistema. Como se puede apreciar, existen dos componentes claramente diferenciados: por un lado el servidor de códigos *voucher* y por otro el componente de autenticación de emergencia del portal Joomla! Como se ha descrito anteriormente, ambos se comunican con el gestor de datos.

En este punto es preciso resaltar que ambos componentes, servidor de códigos *voucher* y componente de autenticación de emergencia, son las dos piezas fundamentales del sistema sobre las que se basará la especificación de requisitos. Asimismo serán la clave sobre la que se profundizará en los siguientes capítulos de esta memoria.

El componente de autenticación de emergencia se despliega en la instalación de Joomla! y su función es la de validar el usuario, contraseña y código *voucher* permitiendo al usuario recuperar el acceso a su cuenta en el portal web. Es necesario que se comunique con el gestor de datos del sistema para verificar si existe una sesión de *vouching* activa y por consiguiente validar el código *voucher*.

El servidor de códigos *voucher* permite por un lado realizar la gestión de los usuarios y por otro la gestión de las sesiones de *vouching*, en la que se incluye la generación del propio código *voucher*.

3.4 Estudio tecnológico

En esta sección se hace un repaso de las tecnologías existentes que por sus características podrían ser candidatas a ser utilizadas en la implementación de la solución propuesta, así como aquellas tecnologías que vienen impuestas por la propia naturaleza del problema.

Como resultado del estudio, se justificará qué tecnologías se adaptan mejor a las necesidades para cubrir con éxito los objetivos del presente proyecto.

3.4.1 Tecnologías impuestas

Joomla! está desarrollado en su gran mayoría en PHP, un lenguaje diseñado originariamente para el desarrollo web de contenido dinámico. Se clasifica dentro de los lenguajes del lado del servidor, lo cual quiere decir que las peticiones son interpretadas por el servidor web dando como resultado una página HTML generada dinámicamente. PHP es uno de los lenguajes más extendidos para el desarrollo web, por su simplicidad y su compatibilidad con la gran mayoría de servidores web y bases de datos del mercado.

Joomla! presenta una arquitectura modular, la cual permite modificar su comportamiento mediante la activación y desactivación de distintos elementos (componentes, módulos y *plug-ins*). También es posible implementar nueva funcionalidad, siempre en PHP y siguiendo una serie de especificaciones, que puede ser integrada de forma sencilla en la aplicación (17).

En Joomla! el proceso de autenticación se lleva a cabo mediante *plug-ins*. Los *plug-ins* son pequeños programas que se ejecutan cuando se lanza el evento correspondiente en la aplicación (17), en este caso cuando el usuario pulsa el botón *login*. Por lo tanto, el componente de autenticación de emergencia constará de un *plug-in* desarrollado en PHP y siguiendo las especificaciones de Joomla! para el desarrollo de este tipo de extensiones. Ésta sería la única dependencia en cuanto a tecnología se refiere, dado que la aplicación gestora de códigos de emergencia no requiere integrarse con ningún otro sistema.

3.4.2 Tecnologías aplicables

En esta sección se realiza un breve análisis de los pros y contras de las distintas tecnologías de servidor existentes para la generación de contenidos web dinámicos que determinarán la elección final.

3.4.2.1 CGI

La tecnología CGI (del inglés *Common Gateway Interface*) permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa, siendo el resultado final de la ejecución objetos MIME. Las aplicaciones que se ejecutan en el servidor reciben el nombre de CGI.

Los CGI se escriben habitualmente en lenguaje Perl, pero también pueden usarse lenguajes como C, C++ o Visual Basic.

CGI es la tecnología más antigua que existe para la generación de páginas web dinámicas de servidor. Además de no mantener el estado (sesión), su principal inconveniente es que para cada usuario que lo ejecuta se crea una nueva instancia, lo cual

lo convierte en una tecnología altamente ineficiente, dado que conlleva un consumo excesivo de recursos de memoria en el servidor.

3.4.2.2 ASP y ASP.NET

ASP (de inglés *Active Server Pages*) es una tecnología del lado del servidor desarrollada por Microsoft para la implementación de sitios web dinámicos. Fue liberado por Microsoft en 1996. Para poder ejecutar las páginas ASP es necesario tener instalado el servidor web IIS (*Internet Information Server*) proprio de Microsoft.

ASP es un lenguaje interpretado, es decir no necesita ser compilado para ejecutarse, lo que lo hace compatible con todos los navegadores web. Además ha tenido siempre una gran aceptación entre los desarrolladores web por la facilidad para su aprendizaje. El código ASP puede ser insertado junto con el código HTML de la página. El lenguaje más utilizado para crear páginas ASP es el VBScript, nativo de Microsoft. Pero también pueden usarse otros lenguajes de script como Perl o JScript entre otros.

En enero de 2002, con la versión 1.0 del *framework* de Microsoft .NET, surge ASP .NET como la tecnología sucesora de ASP. La gran diferencia, es que ASP .NET es un entorno orientado a objetos que admite una amplia variedad de lenguajes y permite hacer uso de todos los recursos ofrecidos por la plataforma .NET.

ASP .NET sólo funciona en entornos Windows con el servidor de Microsoft IIS, lo que supone una desventaja con respecto a otros lenguajes del lado del servidor, como PHP o JSP, ejecutables sobre otros servidores más populares como Apache.

3.4.2.3 Java Servlets

Los Servlets son objetos que se ejecutan en un servidor o contenedor JEE (*Java Enterprise Edition*) y su función es la de generar páginas web de forma dinámica a partir de los parámetros recibidos en la petición. Los Servlets se caracterizan por extender la funcionalidad de una aplicación web que se ejecuta en el servidor.

Una de las grandes ventajas de los Servlets es que pueden ser ejecutados en cualquier servidor web. Éstos se cargan dinámicamente en el servidor cuando se necesitan. Cuando el cliente (navegador) pide una página al contenedor de Servlets (servidor web),

éste delega la petición a un Servlet en particular, el cual procesará los argumentos de la petición y devuelve la respuesta (normalmente una página HTML) al cliente que la solicitó.

Los Servlets son mucho más eficientes y utilizan menos recursos que otras tecnologías de servidor como son los CGI. Sólo existe una copia del Servlet cargada en la máquina virtual y por cada petición se inicia un nuevo hilo, lo cual reduce considerablemente el uso de memoria del servidor y el tiempo de respuesta. Además, tienen persistencia, por lo que permanecen "vivos" una vez terminada la petición.

3.4.2.4 JSP

Es el acrónimo para *Java Server Pages* y fue desarrollado por Sun Microsystems. Es un lenguaje multiplataforma que se ejecuta del lado del servidor, específico para la creación de sitios web dinámicos en Java.

Posee un motor de páginas basado en la tecnología de Servlets de Java. Un JSP consiste en una página HTML con porciones de código Java incrustadas. Esto permite generar código bien estructurado, separando la lógica de la presentación. Las páginas JSP son compiladas en la primera ejecución y convertidas en Servlets, con todas las ventajas que ello conlleva en cuanto a rendimiento se refiere. Para su ejecución se precisa tener instalado un servidor Apache Tomcat.

3.4.2.5 PHP

PHP es un lenguaje de programación interpretado, que originariamente fue diseñado para el desarrollo de contenido web dinámico. El código es interpretado por el servidor web, que debe incorporar un módulo procesador de PHP encargado de genera la página web resultante.

PHP está enfocado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos, ya que es capaz de funcionar con la mayoría de los motores de bases de datos más utilizados en la actualidad, entre los que destaca su gran rendimiento con MySQL.

Está registrado bajo una licencia *Open Source* que permite su libre uso y distribución, es multiplataforma y además está considerado como un lenguaje fácil de

aprender, hechos que han favorecido su gran acogida por la comunidad de desarrolladores. PHP cuenta con una amplia comunidad de usuarios así como una extensa documentación.

3.4.3 Selección de tecnologías aplicables

Una vez analizadas las posibles tecnologías no impuestas aplicables para la implementación de la aplicación gestora de códigos de emergencia, se procede a seleccionar la tecnología que se va a utilizar, incluyendo el razonamiento que ha llevado a tomar dicha decisión.

La primera tecnología en descartarse es CGI, ya que es una tecnología antigua y que en general se considera obsoleta. Debido a su ineficiencia y a la gran cantidad de recursos que consume, su utilización para el presente caso de uso es desaconsejada.

A pesar de que ASP y, más concretamente, su evolución ASP .NET son tecnologías de servidor ampliamente utilizadas hoy en día, su uso viene limitado por el entorno, ya que dependen de una plataforma Microsoft para su ejecución. Es por esto, que también se ha descartado su uso para la implementación del sistema.

PHP, a diferencia de Java, es un lenguaje de tipado débil y, aunque a partir de su versión 5 permite la programación orientada a objetos, es mucho menos restrictivo que Java a la hora de programar, lo que a la larga repercute al desarrollador dificultándole la tarea, especialmente en grandes desarrollos. También dista bastante de Java en cuanto a modularidad se refiere, entendiendo por modularidad la separación en capas definidas en un modelo MVC (Modelo Vista Controlador). La modularidad de un sistema es fundamental de cara a aspectos como la consistencia, la robustez o la mantenibilidad. En PHP es más complejo llevar a cabo dicha modularidad, ya que todas las capas lógicas son implementadas en un mismo archivo .php.

Tradicionalmente Java ha sido considerado un lenguaje más seguro que PHP, aunque bien es cierto se han incluido sustanciales mejoras en las últimas versiones de PHP (18). Los servidores de aplicaciones de Java favorecen la seguridad desde el momento que cada aplicación se ejecuta en su propio contenedor. Para la implementación de la seguridad en portales web, Java cuenta con su propio *framework* de seguridad, además de que existen potentes *frameworks* externos como por ejemplo Spring Security, que hacen transparente

para el desarrollador muchos aspectos de la seguridad, que sí tendría que tener en cuenta a la hora de programar en PHP.

Los estudios demuestran que aunque prácticamente inapreciable, el rendimiento que ofrece la tecnología Java es ligeramente superior a PHP (19). Por lo tanto a mayor rendimiento de la tecnología, mayor escalabilidad de la solución. No obstante, cabe resaltar que el rendimiento de ambas tecnologías es excelente, siendo la diferencia entre ambas tecnologías es mínima, sólo apreciable en situaciones extremas (19).

Por las ventajas que ofrecen tanto los Servlets como las páginas JSP, y por considerarse la tecnología que mejor cubre las necesidades del sistema, se ha concluido que Java será la tecnología emplear.

3.5 Casos de uso

En esta sección se describe como los distintos usuarios (actores) pueden interactuar con el sistema.

Para facilitar la lectura, la descripción de los casos de uso se representará en formato tabla e incluirá los siguientes campos:

- **Título:** Nombre identificativo del caso de uso. Debe ser único y descriptivo.
- **Id:** Identificador unívoco del caso de uso. Debe seguir el siguiente formato: UC_XX, donde:
    - UC hace referencia al término en inglés *"Use Case"*.
    - XX será sustituido por el número de caso de uso. Comenzará en 01 y se incrementará de forma secuencial.
- **Actor:** Agente externo que interactúa con el sistema.
- **Objetivo:** Acción que se persigue tras la interactuación del actor con el sistema.

- **Pre-condición:** Condición que debe cumplirse previamente a la realización del caso de uso.

- **Escenario:** Descripción detallada de los pasos que ha de llevar cabo el actor para completar el caso de uso siguiendo el flujo de ejecución normal.

- **Escenario alternativo:** Descripción detallada de los pasos que ha de llevar a cabo el actor dentro del escenario, en caso de que pueda darse una bifurcación en la ejecución normal del sistema.

- **Post-condición:** Hecho que se cumple si el flujo de eventos del Escenario se ejecuta correctamente.

3.5.1 Definición de actores

Existen tres tipos de usuario que pueden interactuar en el sistema:

- **Solicitante:** Usuario que hace uso del mecanismo de autenticación de emergencia para acceder a su cuenta en el portal de Joomla! mediante el uso del código *voucher* proporcionado por el ayudante.

- **Ayudante:** Usuario que hace uso del servidor de códigos *voucher* para gestionar el acceso del solicitante a su cuenta en el portal de Joomla!

- **Administrador:** Usuario con permisos elevados en el servidor de códigos *voucher* para la realización de tareas de gestión.

3.5.2 Solicitante

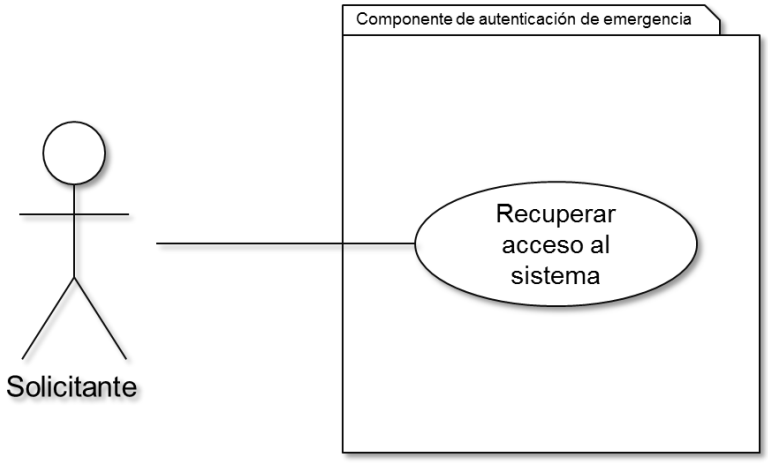

Figura 7. Caso de uso del usuario solicitante

Título	Recuperar acceso al sistema	Versión	1.0	Id	UC-01
Actor	Solicitante				
Objetivo	Autenticarse en el portal de Joomla! cuando no se dispone del método de autenticación primario.				
Pre-condición	▪ Existe conexión a Internet. ▪ El solicitante dispone de una cuenta de usuario en el portal de Joomla! ▪ El solicitante está dado de alta en el servidor de códigos *voucher*. ▪ El solicitante debe disponer de un código *voucher*. ▪ Debe existir una sesión activa asociada a dicho código *voucher*.				
Escenario	1. El actor inicia el portal de Joomla! 2. El actor selecciona la opción de autenticación de emergencia. 3. El sistema muestra el formulario de *login* de emergencia. 4. El actor introduce identificador de usuario, contraseña y código *voucher*. 5. El sistema verifica que el usuario y contraseña son válidos y que existe una sesión de *vouching* activa para el usuario con dicho código *voucher*.				
Esc. alternativo 1	5-4.a. Credenciales incorrectas. 1. Volver al paso 4.				
Post-condición	El actor es autenticado en el portal.				

Tabla 1. Caso de uso UC-01

3.5.3 Ayudante

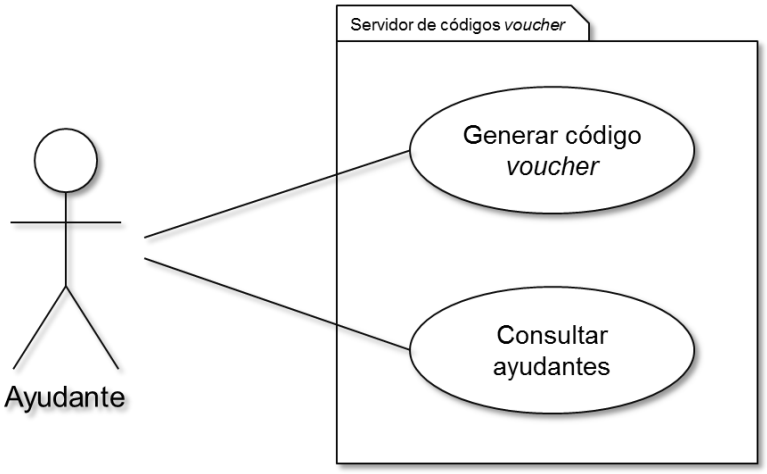

Figura 8. Casos de uso del usuario ayudante

Título	**Generar código *voucher***	Versión	1.0	Id	UC-02
Actor	Ayudante				
Objetivo	Generar un código *voucher* para el solicitante.				
Pre-condición	▪ Existe conexión a Internet. ▪ La base de datos está levantada. ▪ El actor ha iniciado sesión el en servidor de códigos *voucher*. ▪ El solicitante está dado de alta en el servidor de códigos *voucher*. ▪ Existe una relación ayudante-solicitante en el sistema.				
Escenario	1. El actor selecciona la opción de generar código *voucher* 2. El sistema recupera la lista de posibles usuarios solicitantes. 3. El actor selecciona al usuario solicitante. 4. El actor selecciona la opción de generar. 5. El sistema verifica que no existe una sesión activa asociada al solicitante. 6. El sistema genera un código *voucher*. 7. El sistema crea una sesión de *vouching* asociada al solicitante y al ayudante.				
Esc. alternativo 1	5-6a. Existe una sesión activa asociada al solicitante. 1. El sistema muestra un mensaje de error.				
Post-condición	Se visualiza por pantalla el código *voucher* generado.				

Tabla 2. Caso de uso UC-02

Título	Consultar ayudante	Versión	1.0	Id	UC-03
Actor	Ayudante				
Objetivo	Visualizar el usuario que puede actuar como ayudante.				
Pre-condición	▪ Existe conexión a Internet. ▪ La base de datos está levantada. ▪ El actor ha iniciado sesión el en servidor de códigos *voucher*.				
Escenario	1. El actor selecciona la opción de consultar ayudante. 2. El sistema recupera la información asociada al ayudante.				
Post-condición	Se visualiza por pantalla la información básica del ayudante del usuario.				

Tabla 3. Caso de uso UC-03

3.5.4 Administrador

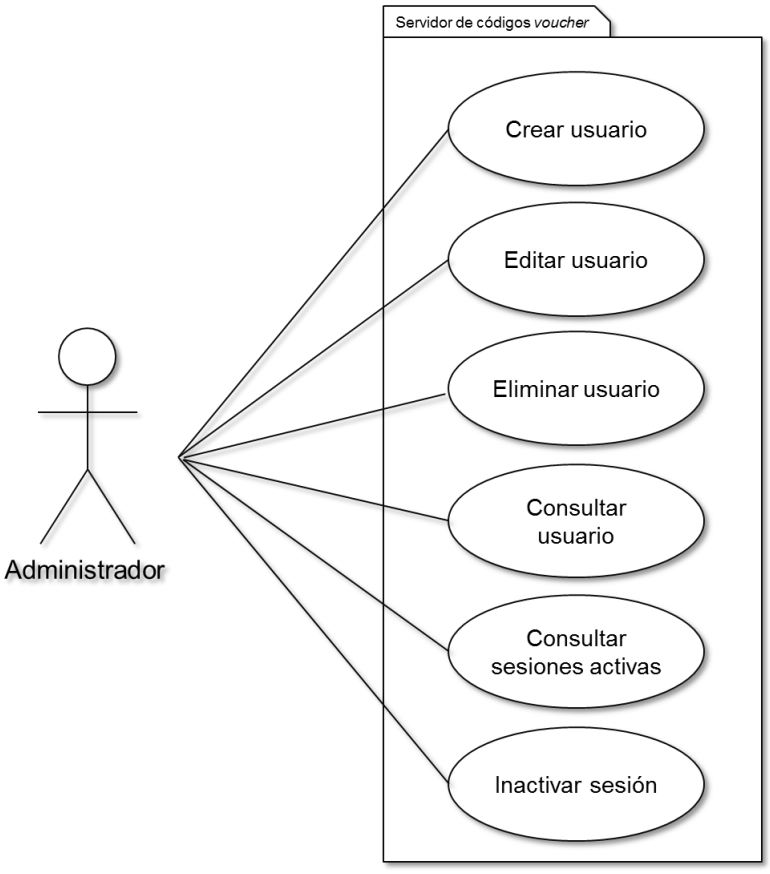

Figura 9. Casos de uso del usuario administrador

Título	Crear usuario	Versión	1.0	Id	UC-04
Actor	Administrador				
Objetivo	Dar de alta a un nuevo usuario en el sistema.				
Pre-condición	▪ Existe conexión a Internet. ▪ La base de datos está levantada. ▪ El actor ha iniciado sesión como administrador en el servidor de códigos *voucher*.				
Escenario	1. El actor seleccionar la opción de crear usuario. 2. El sistema muestra el formulario de alta. 3. El actor introduce los datos en el sistema. 4. El sistema valida que el identificador de usuario introducido no esté en uso. 5. El sistema valida que no hay datos obligatorios vacíos y que el formato de los datos es correcto. 6. El sistema almacena el nuevo usuario. 7. El sistema envía un correo electrónico al nuevo usuario.				
Esc. Alternativo 1	4-5b. Identificador de usuario en uso: 1. Volver al paso 2.				
Esc. Alternativo 2	5-6b. Dato obligatorio vacío o error de formato: 1. Volver al paso 2.				
Post-condición	▪ Se da de alta un nuevo usuario en el sistema. ▪ Se muestra por pantalla el detalle de los datos del usuario.				

Tabla 4. Caso de uso UC-04

Título	Consultar usuarios	Versión	1.0	Id	UC-05
Actor	Administrador				
Objetivo	Visualizar los usuarios del sistema.				
Pre-condición	▪ Existe conexión a Internet. ▪ La base de datos está levantada. ▪ El actor ha iniciado sesión como administrador en el servidor de códigos *voucher*.				
Escenario	1. El actor selecciona la opción de gestionar usuarios. 2. El sistema recupera la lista de usuarios.				
Post-condición	Se visualizan los usuarios dados de alta en el sistema.				

Tabla 5. Caso de uso UC-05

Título	Editar usuario	Versión	1.0	Id	UC-06
Actor	Administrador				
Objetivo	Editar la información de un usuario.				
Pre-condición	▪ Existe conexión a Internet. ▪ La base de datos está levantada. ▪ El actor ha iniciado sesión como administrador en el servidor de códigos *voucher*.				
Escenario	1. El actor selecciona la opción de gestionar usuarios. 2. El actor selecciona al usuario. 3. El actor selecciona la opción de editar. 4. El sistema muestra el formulario con la información del usuario. 5. El actor actualiza la información deseada. 6. El sistema valida que no hay datos obligatorios vacíos y que el formato de los datos es correcto. 7. El sistema actualiza los datos del usuario modificado.				
Esc. alternativo 1	6-7b. Dato obligatorio vacío o error de formato: 1. Volver al paso 4.				
Post-condición	▪ Se actualiza en el sistema la información del usuario. ▪ Se muestra por pantalla el detalle de los datos del usuario.				

Tabla 6. Caso de uso UC-06

Título	Eliminar usuario	Versión	1.0	Id	UC-07
Actor	Administrador				
Objetivo	Dar de baja a un usuario en el sistema.				
Pre-condición	▪ Existe conexión a Internet. ▪ La base de datos está levantada. ▪ El actor ha iniciado sesión como administrador en el servidor de códigos *voucher*.				
Escenario	1. El actor selecciona la opción de gestionar usuarios. 2. El actor selecciona el usuario que se desea eliminar. 3. El actor selecciona la opción de eliminar. 4. El sistema verifica que el usuario se puede eliminar, es decir, que no sea ayudante de otro usuario y que no tenga sesiones de *vouching* activas. 5. El actor confirma la acción eliminar. 6. El sistema elimina al usuario				
Esc. alternativo 1	4-5b. El usuario no se puede eliminar: 1. Volver al paso 1.				
Esc. alternativo 2	5-6b. Cancelar la acción eliminar. 1. Volver al paso 1.				
Post-condición	▪ Se da de baja al usuario en el sistema. ▪ Se visualiza la confirmación de baja de usuario.				

Tabla 7. Caso de uso UC-07

Título	Consultar sesiones activas	Versión	1.0	Id	UC-08
Actor	Administrador				
Objetivo	Visualizar las sesiones de *vouching* activas en el sistema.				
Pre-condición	▪ Existe conexión a Internet. ▪ La base de datos está levantada. ▪ El actor ha iniciado sesión como administrador en el servidor de códigos *voucher*.				
Escenario	1. El actor selecciona la opción de gestionar sesiones. 2. El sistema recupera la lista de sesiones activas.				
Post-condición	Se visualizan las sesiones de *vouching* activas en el sistema.				

Tabla 8. Caso de uso UC-08

Título	Inactivar sesión	Versión	1.0	Id	UC-09
Actor	Administrador				
Objetivo	Marcar como inactiva una sesión de *vouching*.				
Pre-condición	▪ Existe conexión a Internet. ▪ La base de datos está levantada. ▪ El actor ha iniciado sesión como administrador en el servidor de códigos *voucher*. ▪ La sesión de *vouching* está activa.				
Escenario	1. El actor selecciona la opción de gestionar sesiones. 2. El sistema recupera la lista de sesiones activas. 3. El actor seleccionar la sesión que se desea inactivar. 4. El sistema actualiza el estado de la sesión como inactiva.				
Post-condición	▪ La sesión se marca como inactiva. ▪ Se visualizan las sesiones de *vouching* activas en el sistema.				

Tabla 9. Caso de uso UC-09

3.6 Catálogo de requisitos software

En esta sección se recogen los requisitos software que deberán ser cubiertos para cumplir con los objetivos del proyecto. En las siguientes sub-secciones, se recogen los distintos tipos de requisitos definidos por la metodología de la ESA (20). Únicamente se mostrarán los tipos de requisitos que apliquen a esta definición.

La representación de los requisitos de software se ha llevado a cabo mediante una tabla con los siguientes campos:

- **Id:** Identificador único del requisito. Se compone de dos siglas, indicando el tipo de requisito, y un número secuencial.
- **Nombre:** Nombre descriptivo del requisito.
- **Descripción:** explicación detallada del requisito.
- **Estabilidad:** Indica si el requisito es susceptible de ser modificado durante el desarrollo, o si por el contrario no se esperan variaciones. Puede tomar los valores *Alta*, *Media* o *Baja* según corresponda.

- **Prioridad:** Grado de precedencia que se asigna a un requisito para encuadrarlo dentro de la planificación del proyecto. Puede tomar los valores *Alta*, *Media* o *Baja*.

3.6.1 Requisitos Funcionales del sistema

En la siguiente tabla se recogen los requisitos funcionales del sistema a desarrollar. Para todos ellos la fuente es el cliente y por tanto su implementación debe ser obligatoria.

Servidor de códigos *voucher*

Requisitos de Software					
Tipo: Funcional		**Aplicación: Servidor de códigos *voucher***			
Id	**Ver.**	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RF-01	1.0	Iniciar sesión	El sistema debe permitir iniciar sesión a los usuarios.	Alta	Alta
RF-02	1.0	Cerrar sesión	El sistema debe permitir cerrar sesión a los usuarios.	Alta	Alta
RF-03	1.0	Generar código *voucher*	El sistema debe permitir a un usuario generar un código *voucher* para el usuario seleccionado.	Alta	Alta
RF-04	1.0	Consultar ayudantes	El sistema debe permitir a un usuario consultar quién es su ayudante.	Alta	Media
RF-05	1.0	Crear sesión de *vouching*	El sistema debe crear una nueva sesión de *vouching* cuando se genera un código voucher. La sesión de vouching debe tener la siguiente información asociada: ▪ Identificador de sesión. ▪ Identificador del usuario solicitante. ▪ Identificador del usuario ayudante. ▪ Identificador de usuario de Joomla! (del solicitante) ▪ Código voucher. ▪ Activa (Sí/No). ▪ Fecha y hora de creación. ▪ Fecha y hora de expiración.	Alta	Alta

Requisitos de Software					
Tipo: Funcional		**Aplicación: Servidor de códigos *voucher***			
Id	**Ver.**	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RF-06	1.0	Alta de usuario.	El sistema debe permitir al usuario administrador dar de alta a un usuario con la siguiente información: ▪ Identificador de usuario (obligatorio) ▪ Contraseña (obligatorio) ▪ Nombre (obligatorio) ▪ Apellidos (obligatorio) ▪ Dirección de correo electrónico (obligatorio) ▪ Teléfono (obligatorio) ▪ Identificador de usuario de Joomla! (obligatorio) ▪ Usuario ayudante (obligatorio) ▪ Administrador (Sí/No)	Alta	Alta
RF-07	1.0	Actualizar información de usuario.	El sistema debe permitir al usuario administrador actualizar la información de un usuario almacenada en la base de datos.	Alta	Alta
RF-08	1.0	Dar de baja usuario.	El sistema debe permitir al usuario administrador dar de baja a un usuario, eliminándolo de la base de datos.	Alta	Alta
RF-09	1.0	Enviar e-mail de alta.	El sistema debe enviar un correo electrónico de bienvenida al usuario, cuando éste es dado de alta en el sistema.	Alta	Alta

Requisitos de Software					
Tipo: Funcional		**Aplicación: Servidor de códigos *voucher***			
Id	**Ver.**	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RF-10	1.0	Consultar usuarios.	El sistema debe de permitir al usuario administrador consultar la lista de usuarios dados de alta. Para cada usuario se deben mostrar los siguientes datos: ▪ Identificador de usuario ▪ Nombre completo (Nombre y apellidos) ▪ Dirección de correo electrónico ▪ Teléfono	Alta	Alta
RF-11	1.0	Consultar sesiones activas.	El sistema debe permitir al usuario administrador consultar la lista de sesiones activas en el sistema. Para cada sesión se deben mostrar los siguientes datos: ▪ Identificador de sesión. ▪ Identificador del usuario solicitante. ▪ Identificador del usuario ayudante. ▪ Activa (Sí/No). ▪ Fecha y hora de creación. ▪ Fecha y hora de expiración.	Alta	Media
RF-12	1.0	Inactivar sesión.	El sistema debe permitir al usuario administrador marcar como inactiva una sesión activa.	Media	Media
RF-13	1.0	Modificar idioma.	El sistema debe permitir al usuario administrador seleccionar el idioma por defecto de la aplicación.	Alta	Alta
RF-14	1.0	Modificar la longitud del código *voucher*.	El sistema debe permitir al usuario administrador configurar la longitud del código *voucher*.	Alta	Media

Requisitos de Software					
Tipo: Funcional		**Aplicación: Servidor de códigos *voucher***			
Id	**Ver.**	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RF-15	1.0	Establecer la validez de la sesión de *vouching*.	El sistema debe permitir al usuario administrador configurar el periodo de validez de la sesión de *vouching*.	Alta	Alta

Tabla 10. Requisitos funcionales del servidor de códigos voucher

Componente de autenticación de emergencia

Requisitos de Software					
Tipo: Funcional		**Aplicación: Componente de autenticación de emergencia**			
Id	**Ver.**	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RF-16	1.0	Validar usuario y contraseña.	El sistema debe validar que el usuario y contraseña introducidos son correctos.	Alta	Alta
RF-17	1.0	Validar código *voucher*.	El sistema debe validar que el código voucher introducido es válido. Para ello, el sistema comprobará que existe una sesión de *vouching* asociada a dicho código *voucher* cuyo estado es "Activo" y no se ha superado la fecha de expiración.	Alta	Alta

Tabla 11. Requisitos funcionales del componente de autenticación de emergencia

3.6.2 Requisitos inversos del sistema

En esta sección se recogen aquellos requisitos que reflejan aquello que la aplicación no debe hacer. El origen de estos requisitos es el equipo de desarrollo.

Cabe resaltar que la descripción de los requisitos inversos no requiere especificar la prioridad.

Servidor de códigos *voucher*

Requisitos de Software					
Tipo: Inverso		**Aplicación: Servidor de códigos *voucher***			
Id	**Ver.**	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RI-01	1.0	No repetir el mismo identificador de usuario.	El sistema no debe permitir dar de alta con un identificador de usuario ya existente.	Alta	-
RI-02	1.0	No modificar identificador de usuario.	El sistema no debe permitir editar el identificador de usuario.	Alta	-
RI-03	1.0	Usuarios sin ayudantes.	El sistema no debe permitir que un usuario no tenga ningún ayudante asignado en el sistema. Al dar de baja a un usuario el sistema debe comprobar que ninguno de los solicitantes asociados a ese usuario se queda sin ayudante.	Alta	-
RI-04	1.0	No tener más de una sesión de *vouching* activa.	El sistema no debe permitir generar un código voucher para un usuario que tiene asociada una sesión de *vouching* en estado activo y que no ha expirado.	Alta	-

Tabla 12. Requisitos inversos del servidor de códigos voucher

Componente de autenticación de emergencia

Requisitos de Software					
Tipo: Inverso		**Aplicación: Componente de autenticación de emergencia**			
Id	**Ver.**	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RI-05	1.0	No repetir el mismo código *voucher*.	El sistema no debe permitir a un usuario autenticarse dos veces con un mismo código *voucher*.	Alta	-

Tabla 13. Requisitos inversos del componente de autenticación de emergencia

3.6.3 Requisitos no funcionales del sistema

En esta sección se presentan los requisitos no funcionales, que complementan a los requisitos funcionales, y hacen relación a las restricciones en el diseño y la implementación. La fuente de todos ellos es el equipo de desarrollo.

Servidor de códigos *voucher*

Requisitos de Software					
Tipo: Interfaz		**Aplicación: Servidor de códigos *voucher***			
Id	**Ver.**	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RNF-01	1.0	Internacionalización de la interfaz.	La interfaz de usuario deberá poder visualizarse en distintos idiomas: español e inglés.	Alta	Alta
RNF-02	1.0	Visualización de las distintas funcionalidades.	Las funciones del sistema permitidas para el usuario deben estar visibles en todo momento y deberán ser accesibles mediante un *click*.	Alta	Alta
RNF-03	1.0	Misma estructura de páginas.	La interfaz de usuario deberá presentar la misma estructura de páginas en toda la aplicación.	Alta	Alta
Tipo: Manejo de errores		**Aplicación: Servidor de códigos *voucher***			
Id	**Ver.**	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RNF-04	1.0	Validación de formato de los campos de los formularios.	El sistema debe validar que los datos introducidos en los formularios tienen un formato válido. En caso de no ser así, se debe mostrar el correspondiente error.	Alta	Alta
RNF-05	1.0	Validación de obligatoriedad de los campos de los formularios.	El sistema debe validar que se han introducido todos los datos obligatorios en el formulario. De no ser así se debe mostrar el correspondiente mensaje de error.	Alta	Alta
RNF-06	1.0	Visualización de mensaje de error por acciones no permitidas.	El sistema siempre debe mostrar un mensaje de error en caso de realizar una actividad no permitida en el sistema.	Alta	Alta
Tipo: Seguridad		**Aplicación: Servidor de códigos *voucher***			
Id	**Ver.**	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**

RNF-07	1.0	El sistema debe disponer de un método de autenticación de doble factor.	Para poder hacer uso del sistema los usuarios deberán autenticarse previamente mediante usuario, contraseña y un código OTP. La aplicación generadora de códigos OTP será *Google Authenticator*.	Alta	Alta
RNF-08	1.0	Almacenamiento de las contraseñas cifradas.	El sistema debe almacenar únicamente la función resumen de las contraseñas en la base de datos.	Alta	Alta
RNF-09	1.0	Aleatoriedad del código *voucher*.	La función de generación de códigos *voucher* debe ser criptográficamente segura y con un alto grado de aleatoriedad.	Alta	Alta
Tipo:		**Aplicación: Servidor de códigos *voucher***			
Id	**Ver.**	**Nombre**	**Descripción**	**Estabilidad**	**Prioridad**
RNF-10	1.0	Multiplataforma	La aplicación debe ser multiplataforma.	Alta	Alta
RNF-11	1.0	Compatibilidad con navegadores	La aplicación debe visualizarse correctamente en los navegadores de uso más frecuente: Internet Explorer, Firefox, Google Chrome y Safari.	Alta	Alta

Tabla 14. Requisitos no funcionales del servidor de códigos voucher

3.7 Pruebas de aceptación

En esta sección se definen las pruebas de aceptación, cuyo principal objetivo es el de determinar el grado de calidad del software una vez desarrollado.

En la siguiente tabla se recoge el conjunto de casos de prueba que permitirán revisar los requisitos definidos en la fase de análisis y verificar su correcta implementación en el sistema.

Las pruebas aquí definidas se ejecutarán al completo una vez finalizada la fase de implementación. Si bien, durante el desarrollo se irán realizando pruebas unitarias del software que se vaya implementado con el fin de identificar los posibles errores en fases más tempranas. Si la ejecución de las pruebas de aceptación no es satisfactoria, se volverá a ejecutar una vez se hayan resuelto los errores. Este proceso se repetirá hasta que el resultado del 100% de los casos de prueba sea positivo.

Servidor de códigos *voucher*

Pruebas de aceptación				
Id	**Ver.**	**Requisitos**	**Entrada**	**Salida**
PA-01	1.0	RF01	1. Iniciar la aplicación. 2. Introducir identificador de usuario "maria" y contraseña "passwd". 3. Introducir la clave temporal proporcionada por el generador de claves de *Google Authenticator*.	El usuario "María García" es autenticado en el sistema.
PA-02	1.0	RF02	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Seleccionar la opción de cerrar sesión.	La sesión se cierra.
PA-03	1.0	RF03, RF-05	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Pulsar sobre la opción de generar código *voucher*. 3. Seleccionar de la lista al solicitante "Juan López". 4. Pulsar sobre la opción de generar.	Se muestra por pantalla el código *voucher* generado. Se crea una nueva sesión de *vouching* en el sistema.
PA-04	1.0	RF03, RI-04	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Pulsar sobre la opción de generar código *voucher*. 3. Seleccionar de la lista al solicitante "Juan López". 4. Pulsar sobre la opción de generar.	No se genera un nuevo código *voucher* ni se crea una nueva sesión de *vouching* en el sistema, debido a que para el solicitante seleccionado ya existe una sesión activa. En su lugar, se muestra por pantalla un mensaje de error.
PA-05	1.0	RF04	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Pulsar sobre la opción de consultar ayudante.	Se muestra por pantalla la información del ayudante "José Martín", ayudante del usuario "maria".

Pruebas de aceptación				
Id	**Ver.**	**Requisitos**	**Entrada**	**Salida**
PA-06	1.0	RF-06, RF-09	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Pulsar sobre la opción de crear usuario. 3. Introducir la información requerida para el alta. 4. Pulsar sobre la opción de crear.	El usuario se crea en el sistema. Se envía un correo electrónico de bienvenida al usuario dado de alta.
PA-07	1.0	RF-06, RI-01	1. Seguir los pasos 1-3 de la prueba PA-06. 2. En el paso 2 introducir el identificador de usuario "juan".	El usuario no se crea en el sistema puesto que el identificador de usuario introducido ya está en uso. Se muestra por pantalla un mensaje de error.
PA-08	1.0	RF-10	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Pulsar sobre la opción de consultar usuarios.	Se muestra por pantalla la lista de usuarios dados de alta en el sistema.
PA-09	1.0	RF-07, RI-02	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Pulsar sobre la opción de consultar usuarios. 3. Seleccionar al usuario "Juan López". 4. Pulsar sobre la opción de editar usuario. 5. Modificar la información a actualizar. 6. Pulsar sobre la opción de guardar.	La información del usuario se actualizada en el sistema. El campo identificador de usuario aparece como no editable.
PA-10	1.0	RF-08	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Pulsar sobre la opción de consultar usuarios. 3. Seleccionar al usuario "Ana Sánchez" 4. Pulsar sobre la opción de eliminar usuario.	El usuario es eliminado del sistema.
PA-11	1.0	RF-08, RI-03	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Seleccionar al usuario "José Martín" 3. Pulsar sobre la opción de eliminar usuario.	El usuario no es eliminado del sistema puesto que es ayudante de otro usuario. Se muestra por pantalla un mensaje de error.

Pruebas de aceptación				
Id	**Ver.**	**Requisitos**	**Entrada**	**Salida**
PA-12	1.0	RF-11	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Pulsar sobre la opción de consultar sesiones.	Se muestra por pantalla la lista de sesiones existentes en el sistema en estado activo y que no hayan expirado.
PA-13	1.0	RF-12	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Pulsar sobre la opción de consultar sesiones. 3. Seleccionar una sesión. 4. Pulsar sobre la opción de cancelar.	Se marca la sesión como inactiva.
PA-14	1.0	RF-13	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Pulsar sobre la opción configuración. 3. Seleccionar otro idioma de la lista. 4. Pulsar sobre la opción de guardar.	La interfaz de usuario de la aplicación se visualiza en el idioma seleccionado.
PA-15	1.0	RF-14	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Pulsar sobre la opción configuración. 3. Modificar la longitud del código voucher a 8 caracteres. 4. Pulsar sobre la opción de guardar. 5. Generar un nuevo código voucher (PA-03)	El nuevo código voucher generado tiene la longitud indicada.
PA-16	1.0	RF-15	1. Seguir los pasos 1-3 de la prueba PA-01. 2. Pulsar sobre la opción configuración. 3. Modificar el tiempo de validez de la sesión de *vouching* a 10 minutos. 4. Pulsar sobre la opción de guardar. 5. Generar un nuevo código voucher (PA-03)	La sesión de *vouching* generada tiene un periodo de validez igual al tiempo indicado.

Tabla 15. Pruebas de aceptación del servidor de códigos voucher

Componente de autenticación de emergencia

Pruebas de aceptación				
Id	**Ver.**	**Requisitos**	**Entrada**	**Salida**
PA-17	1.0	RF-16, RF-17	1. Acceder al portal web. 2. Pulsar sobre la opción de no disponer de la clave secreta. 3. Introducir un usuario "juan" y contraseña "pass123". 4. Introducir un código voucher válido.	El usuario es autenticado en el portal web.
PA-18	1.0	RF-16, RF-17, RI-05	1. Seguir los pasos 1 -3 de la prueba PA-17. 2. Introducir un código voucher ya utilizado con anterioridad.	El usuario no es autenticado en el portal web. Se muestra por pantalla un mensaje de error.
PA-19	1.0	RF-16, RF-17, RI-06	1. Seguir los pasos 1 -3 de la prueba PA-17. 2. Introducir un código voucher asociado a una sesión de *vouching* expirada.	El usuario no es autenticado en el portal web. Se muestra por pantalla un mensaje de error.

Tabla 16. Pruebas de aceptación del componente de autenticación de emergencia

Capítulo 4

Diseño detallado

Una vez llevado a cabo el análisis, en este capítulo se recogen todos los aspectos relacionados con el diseño del software de la solución. Primeramente, en la sección 4.1 se pone de manifiesto la utilización del patrón de diseño Modelo Vista Controlador, tanto para el componente de autenticación de emergencia como para el servidor de códigos *voucher*. Además se introducen los *frameworks* que se utilizarán para llevar a cabo el desarrollo del servidor de códigos *voucher* y que, inevitablemente, afectarán a la estructura de la aplicación. En la sección 4.2 se presenta la arquitectura de componentes definitiva del sistema detallando la funcionalidad y estructura de clases de cada uno de ellos. Por último, en la sección 4.3 se incluyen los diagramas de secuencia de cada uno de los casos de uso mostrando las interacciones entre los distintos componentes que conforman el sistema.

4.1 Patrones de diseño del software

Debido a la magnitud del proyecto y dado que el uso de patrones de diseño se considera una buena práctica de programación se ha optado por utilizar el patrón de diseño Modelo Vista Controlador, en adelante MVC.

El patrón MVC se caracteriza por separar el acceso a los datos, la lógica de aplicación y la presentación de los datos. De esta manera, se obtiene un código fuente más estructurado y por consiguiente más fácil de mantener (21). Éste se descompone en tres niveles:

- **El modelo:** está compuesto por una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

- **La vista:** es la interfaz de usuario. Compone la información que se envía al cliente y los mecanismos interacción con éste.

- **El controlador:** es la capa encargada de manejar y responder las solicitudes del usuario. Actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y llevando a cabo las transformaciones necesarias para adaptar los datos a las necesidades de cada uno.

El ciclo de vida del patrón MVC comienza cuando el usuario interactúa con la interfaz de usuario y realiza una acción (por ejemplo, pulsa un botón o un enlace), la cual desencadena una solicitud al controlador. El controlador recibe la acción solicitada por el usuario y comienza la gestión de dicha solicitud. El controlador accede al modelo, actualizándolo en caso necesario, y facilita la información necesaria a la vista. La vista genera la interfaz apropiada a mostrar al usuario y actualiza en ella la información proporcionada por el controlador. Por último, la interfaz de usuario queda a la espera de nuevas interacciones del usuario lo que daría lugar a un nuevo ciclo (21).

El patrón MVC presenta numerosas ventajas, como por ejemplo facilitar la labor de agregar nuevas formas de representación de los datos sin afectar por ello a la lógica de negocio. También permite modificar los objetos que conforman la lógica de negocio, bien sea para mejorar el rendimiento o para migrar a otra tecnología. Además, posibilita la reutilización de los distintos componentes. De cara a la gestión del tiempo, hace el

desarrollo más eficiente, permitiendo paralelizar tareas entre diferentes equipos. Por último resaltar que simplifica la depuración de errores y produce desarrollos más escalables.

Sin embargo, el patrón MVC también presenta algunos inconvenientes. Por un lado, el número de archivos a desarrollar y mantener se incrementa notablemente, por otro, la curva de aprendizaje se incrementa en comparación con otros modelos más sencillos, y en muchos casos es necesaria o aconsejable la utilización de un *framework* que guíe al desarrollador.

A pesar de esto, el patrón MVC ha demostrado probada eficacia a lo largo de los años en el desarrollo de arquitecturas consistentes, reutilizables y más fáciles de mantener y ha sido ampliamente adoptado por la comunidad de desarrolladores web.

Joomla! está desarrollado siguiendo el patrón MVC y como tal, es preciso que las extensiones, para que puedan ser instaladas y funcionar correctamente, también sean implementadas siguiendo este mismo patrón. En cuanto al servidor de códigos *voucher*, se ha decido utilizar el patrón MVC para su diseño, dado que su uso está altamente recomendado en el desarrollo de aplicaciones web.

4.1.1 Frameworks

La implementación del patrón MVC para el servidor de códigos voucher se ha llevado a cabo con el *framework* Spring MVC junto con *Hibernate* para la capa de persistencia de los datos.

Spring es un *framework* ligero especialmente diseñado para el desarrollo de aplicaciones Java. Al estar desarrollado bajo una licencia *Open Source*, puede ser utilizado sin coste alguno. *Spring* tiene la gran ventaja de que simplifica notablemente el desarrollo de aplicaciones web, sin embargo la curva de aprendizaje es acusada al inicio.

Hibernate es una herramienta Java ORM (del inglés *Object Relational Model*) que facilita el mapeo de atributos de cualquier tipo de una base de datos relacional y el modelo de objetos de una aplicación usando archivos XML para declarar las relaciones.

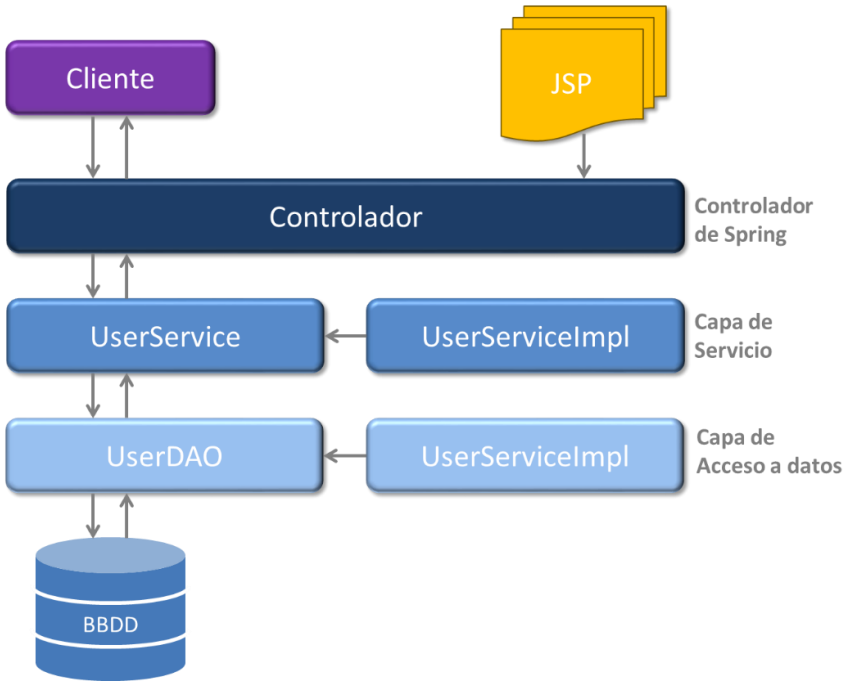

Figura 10. Estructura de una aplicación web implementada con Spring

Las aplicaciones implementadas con Spring están compuestas por tres capas, tal y como se puede apreciar en la Figura 10. Cuando el controlador determina que se debe realizar alguna acción sobre la base de datos, éste lanza una petición a la capa de servicio. La capa de servicio a su vez invoca a la capa de acceso a datos, la cual interacciona directamente con la base de datos mediante el API de *Hibernate*. Ambas capas, de servicio y de acceso a datos, conforman la capa del modelo. Por su parte, la capa de la vista está formada por los JSP (22).

4.2 **Diseño del software**

En la siguiente figura, se muestra la estructura de componentes definitiva del sistema:

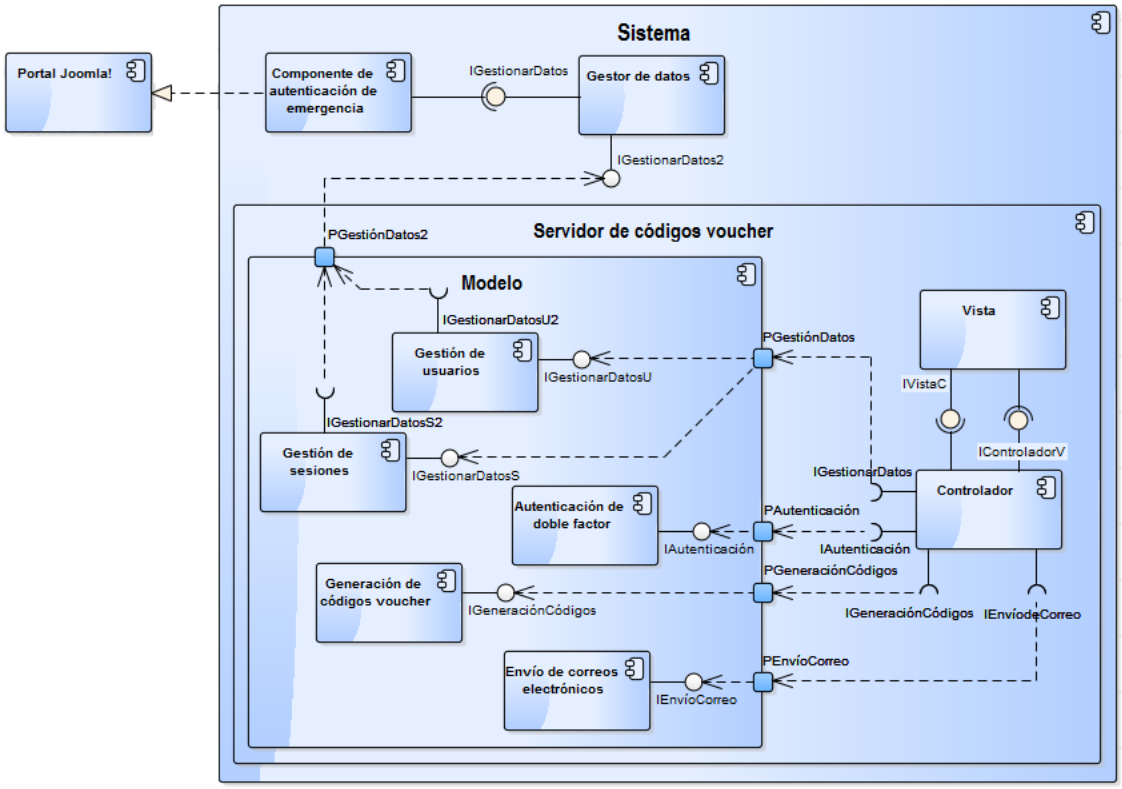

Figura 11. Arquitectura de componentes definitiva del sistema

A continuación se describen en mayor profundidad los distintos componentes definidos durante la fase de análisis, haciendo especial énfasis en su implementación.

Con el único objetivo de facilitar la lectura de las siguientes subsecciones, se han obviado aquellas variables, métodos y clases que por sus características carecen de interés para la implementación, como por ejemplo variables auxiliares y los métodos *get* y *set* en determinadas clases.

4.2.1 Componente de autenticación de emergencia

Este componente permite a un usuario autenticarse en el portal de Joomla! mediante la introducción de su identificador de usuario, contraseña y código *voucher*.

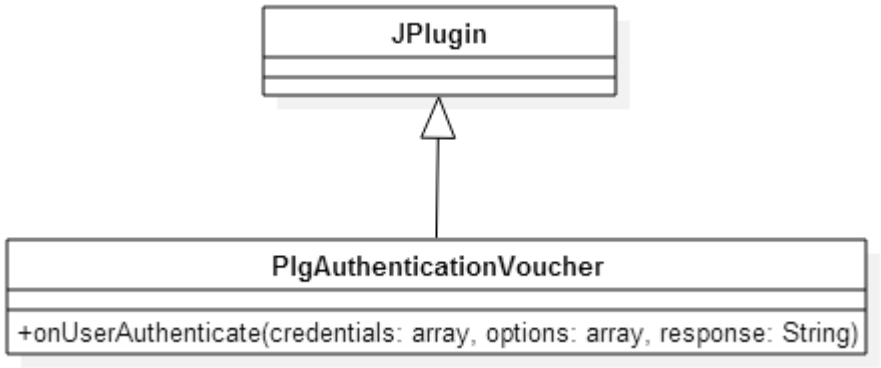

Figura 12. Diagrama de clases del componente de autenticación de emergencia

El *plug-in* de autenticación de emergencia extiende la clase JPlugin del API de Joomla! Los *plug-ins* generalmente tienen un único método que se corresponde con un evento. Cuando éste se produce, se ejecuta el método definido en el *plug-in*.

El método *onUserAuthenticate* valida contra la base de datos de Joomla! usuario y contraseña. Además contiene la lógica de negocio necesaria para consultar el estado de la sesión de *vouching* en la base de datos del sistema y así validar el código *voucher* introducido por el usuario.

4.2.2 Componente “Autenticación Google”

El objetivo de este componente consiste en proporcionar los servicios necesarios que hacen posible la autenticación de doble factor, usando como factor de posesión *Google Authenticator*, para acceder a la aplicación.

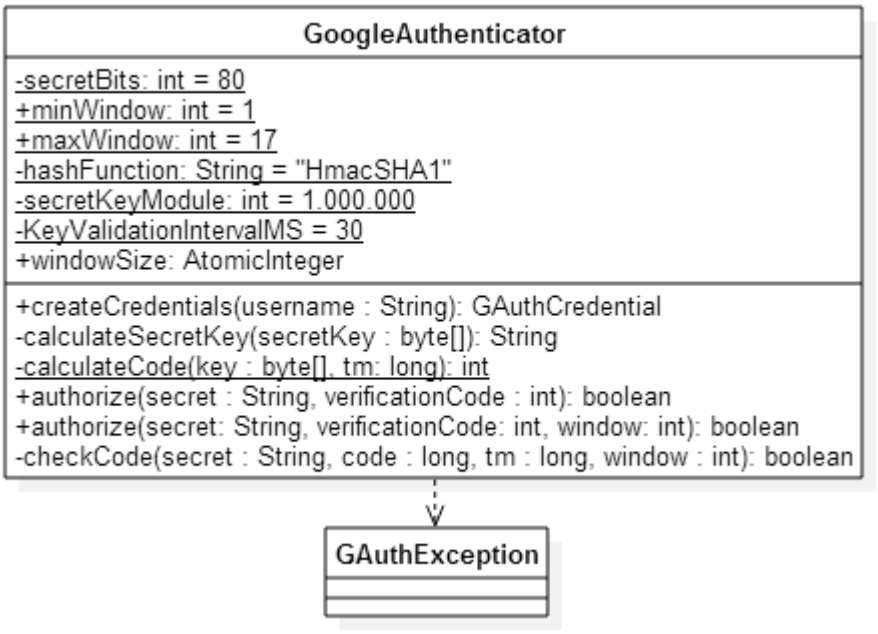

Figura 13. Diagrama de clases del componente "Google Authenticator"

Este componente está formado por la clase *GoogleAuthenticator*, cuyos servicios son consumidos por el componente "Controlador", tanto en el momento de dar de alta un usuario, para generar su clave secreta como cada vez que un usuario se autentica en la aplicación:

El componente proporciona la siguiente funcionalidad:

- *calculateSecretKey*: este método genera una clave secreta que es asignada al usuario, la cual le será requerida por la aplicación de *Google Authenticator* para poder configurar la cuenta asociada al servidor de códigos *voucher*. Una vez introducida, se comienzan a generar las claves OTP válidas que posibilitarán al usuario acceder al servidor de códigos. Además esta clave también es utilizada por el método *authorize* que valida si la clave OTP introducida en el formulario de inicio de sesión es válida para dicho usuario.

- *authorize*: hace uso del método privado *checkCode* para validar que la clave OTP introducida es válida, con respecto a una clave secreta y dado un instante de tiempo.

- *checkCode*: método que implementa la RFC 6238 (23) que corresponde al algoritmo de contraseña de un solo uso basado en el tiempo o TOTP.

La clase *GAuthException* extiende de la clase java.lang.RuntimeException y define las excepciones propias de los servicios de *GoogleAuthenticator*.

4.2.3 Componente "Generador de códigos de emergencia"

Como su propio nombre indica, es el componente encargado de la generación de los códigos de emergencia. Su función es crear cadenas alfanuméricas aleatorias de la longitud especificada. Para ello hace uso de la clase *SecureRandom*, presente en el paquete *java.security*, la cual se caracteriza por generar números pseudo-aleatorios que pueden ser considerados seguros desde un punto de vista criptográfico.

Los caracteres que pueden aparecer en el código de emergencia vienen definidos por la constante *symbols*.

Figura 14. Diagrama de clases del componente "Generador de códigos de emergencia"

4.2.4 Componente "Envío de correos"

Este componente encapsula la funcionalidad necesaria para realizar el envío de correos electrónicos desde la aplicación en formato MIME. Los correos electrónicos se generan en el idioma por defecto de la aplicación.

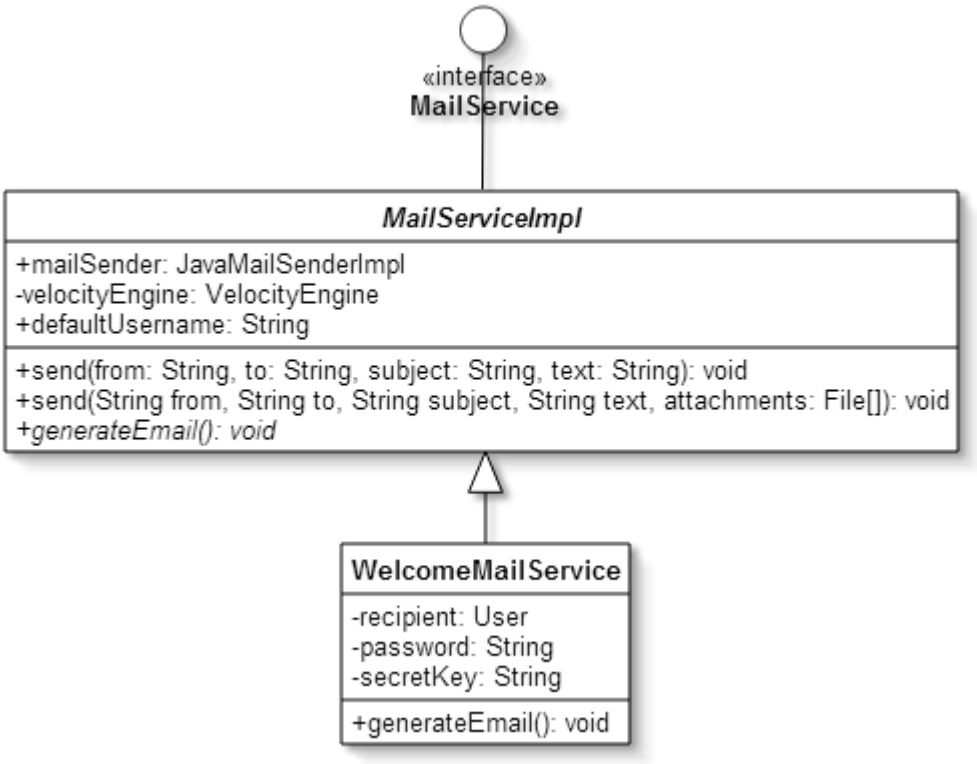

Figura 15. Diagrama de clases del componente "Envío de correos"

Inicialmente, la aplicación envía un correo electrónico de bienvenida cuando un usuario ha sido dado de alta, en el cual se informa al usuario de su nombre de usuario, su contraseña y la clave secreta necesaria para la configuración de .

Para simplificar en un futuro la inclusión de nuevos envíos de correo electrónico desde la aplicación, se ha empleado el patrón de diseño *"Strategy"*. Este patrón tiene la particularidad de permitir que distintos objetos con un comportamiento similar compartan ese comportamiento común e implementen aquella funcionalidad propia de cada uno (21). La clase abstracta *MailServiceImpl* actúa como *estrategia*, declarando una interfaz común para el envío de correos electrónicos en formato MIME. Cada *estrategia concreta*, en este caso *WelcomeMailService*, implementa la composición del mensaje, utilizando para ello su propia plantilla en el idioma seleccionado y sus propios atributos.

4.2.5 Componente “Modelo”

Este componente contiene una representación de los datos que maneja la aplicación, su lógica de negocio y sus mecanismos de persistencia. Está organizado en una arquitectura de tres niveles: la capa de representación de datos (o el modelo propiamente dicho), la capa de acceso a los datos o de persistencia (también conocida como DAO, acrónimo del inglés *Data Access Object*) y la capa de servicios que encapsula el acceso a los datos y que actúa de interfaz para el controlador.

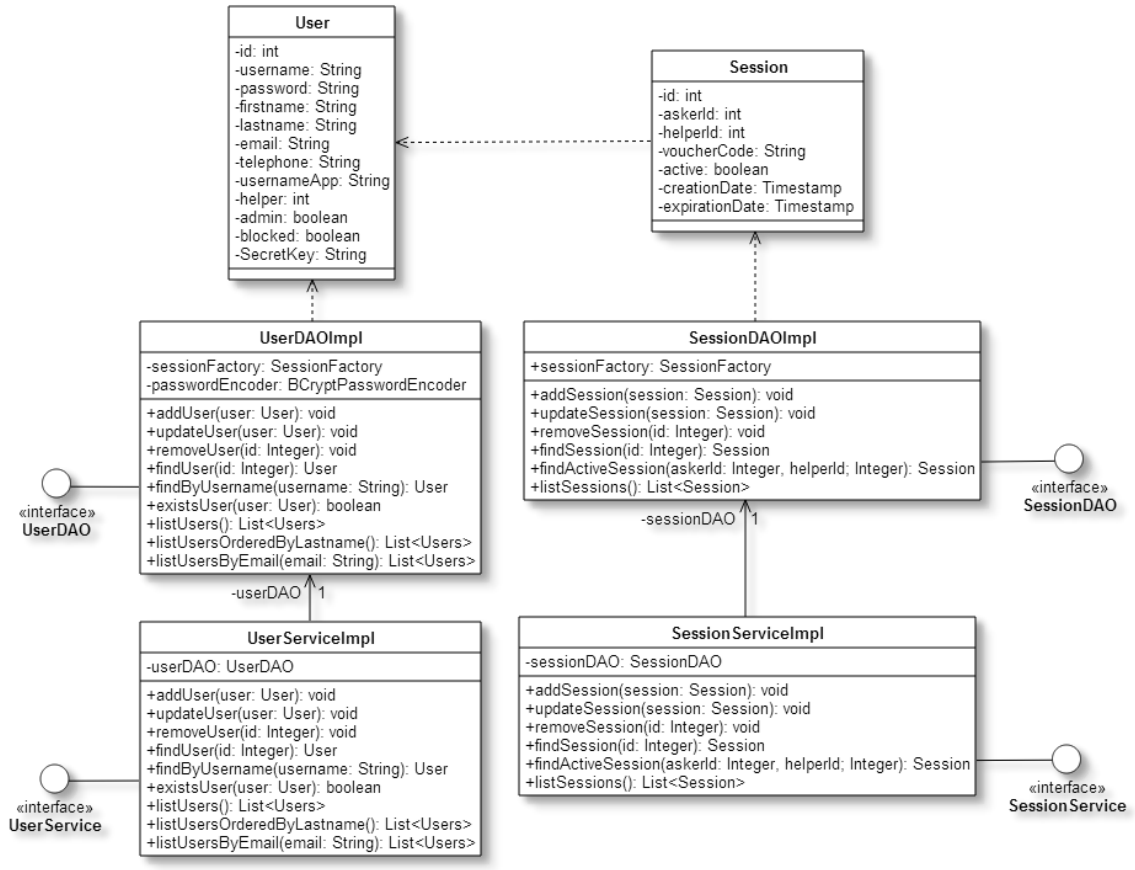

Figura 16. Diagrama de clases del componente “Modelo”

La aplicación maneja dos tipos de datos: usuarios y sesiones, cada uno de ellos respaldado por una clase que lo representa denominada *Bean*. De acurdo al modelo impuesto por *Hibernate*, cada *Bean* se corresponde con una tabla de la base de datos y cada atributo con una columna. Aunque no se han representado en el diagrama por razones de

legibilidad, los *Beans* implementan los métodos de asignación y recuperación para cada uno de los atributos.

La capa de persistencia contiene un objeto de la clase *SessionFactory* de *Hibernate*, encargada de abrir y gestionar sesiones de *Hibernate* con la base de datos. Sigue el patrón de diseño *Singleton*, puesto que sólo existe una instancia en de este objeto por aplicación y base de datos. Ésta se inicializa en el fichero de configuración XML que se carga al iniciarse la aplicación y posteriormente se recupera tantas veces es necesario para operar en la base de datos. En el caso del usuario, se ha incluido además el atributo *passwordEncoder*, utilizado para obtener la función *hash* de la contraseña previamente a ser almacenada en la base de datos, por motivos de seguridad.

La capa de servicio meramente encapsula la funcionalidad proporcionada por la capa de persistencia en forma de servicios que son consumidos por el controlador.

4.2.6 Componente “Vista”

El objetivo de este componente es proporcionar la interfaz de usuario de la aplicación. Está compuesto por una serie de JSP que generan las distintas pantallas que conforman la aplicación.

4.2.7 Componente “Controlador”

El componente Controlador de la aplicación gestora de códigos de emergencia, se compone de una única clase. Esta clase es la encargada de definir todo el flujo de la aplicación.

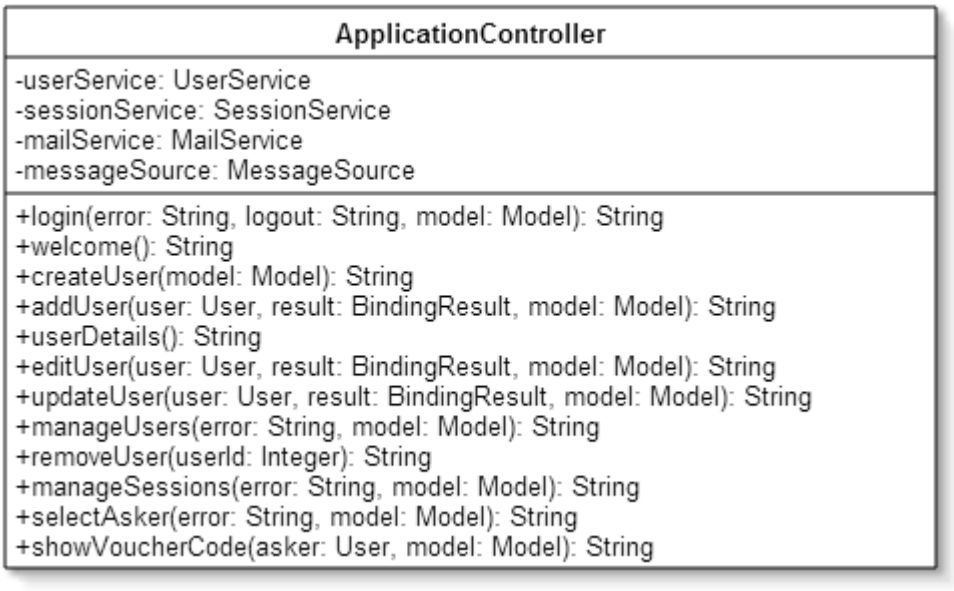

Figura 17. Diagrama de clases del componente "Controlador"

Cada uno de los métodos presente en el controlador está mapeado contra una URL. Cuando el usuario lleva a cabo una acción a través de la vista ésta se traduce en una URL que debe tener su correspondiente método asignado en el controlador. Ahí es donde reside el grueso de la lógica de negocio que es capaz de determinar cuál será la siguiente tarea a llevar a cabo.

4.3 **Diagramas de secuencia**

Una vez descritas las clases que forman cada uno de los componentes del sistema, en esta sección se muestra la interacción llevada a cabo entre dichas clases en la consecución de los casos de uso definidos en capítulo de análisis. Esta interacción entre clases, se presentará a través de diagramas de secuencia.

De cara a simplificar los diagramas, no se representan los escenarios alternativos de los casos de uso. Tampoco se representan aquellas clases propias del funcionamiento interno de la aplicación (a menos que ayude a clarificar el flujo), así como invocaciones a métodos que no aportan información ni son de interés para el diseño.

Al igual que en ocasiones anteriores, esta sección se divide en dos partes: diagramas de secuencia del componente de autenticación de emergencia del portal Joomla! y diagramas de secuencia del servidor de códigos *voucher*.

4.3.1 Diagramas de secuencia del componente de autenticación de emergencia

En esta sección se representa el diagrama de secuencia asociado al caso de uso definido en la sección 3.5.2 de este documento.

UC01 – Recuperar acceso al sistema

La interacción a alto nivel entre las clases del API de Joomla! con el *plug-in* desarrollado que permite a un usuario autenticarse a través del código *voucher*, se presenta en la Figura 18.

Ante la situación de un usuario que quiere acceder a su cuenta en el portal de Joomla!, para lo cual requiere autenticarse mediante usuario, contraseña y clave OTP. Sin embargo, por alguna razón no dispone de su dispositivo generador de claves, por lo que éste es instado a introducir en su lugar el código *voucher* que le fue proporcionado por su ayudante. Una vez introducidas todas las credenciales, el usuario confirma la acción a través de la interfaz de usuario.

La vista indica al controlador que el usuario ha iniciado la acción de *login*, enviándole al mismo tiempo un *array* con las credenciales introducidas. El controlador obtiene la instancia del *JEventDispatcher*, encargado de identificar qué tipo de evento se ha producido y localizar así al *plug-in* adecuado. Mediante el método *trigger*, el controlador indica al *JEventDispatcher* cuál es el evento y el *array* con los parámetros de entrada necesarios para que el *plug-in* pueda llevar a cabo su tarea. El *JEventDispatcher* invoca al método del *plug-in* correspondiente al evento iniciado, *onUserAuthenticate*, el cual se encarga de realizar la autenticación propiamente dicha. Primeramente valida el usuario y contraseña de Joomla! mediante la función *verifyPassword* de la clase *JUserHelper*, que permite comparar la contraseña en claro introducida con el resultado de la función *hash* almacenada en la base de datos. Si el resultado es satisfactorio, se obtiene la sesión de *vouching* del usuario en la base de datos del sistema (distinta de la de Joomla!) y se verifica

el código *voucher* utilizando de nuevo la función *verifyPassword*. Una vez validado, se establece el atributo estado de la clase *JAuthentication*, que determinará de manera oficial que el usuario se encuentra autenticado en el portal. El controlador por último actualiza la vista del usuario mostrando por ejemplo la página de bienvenida.

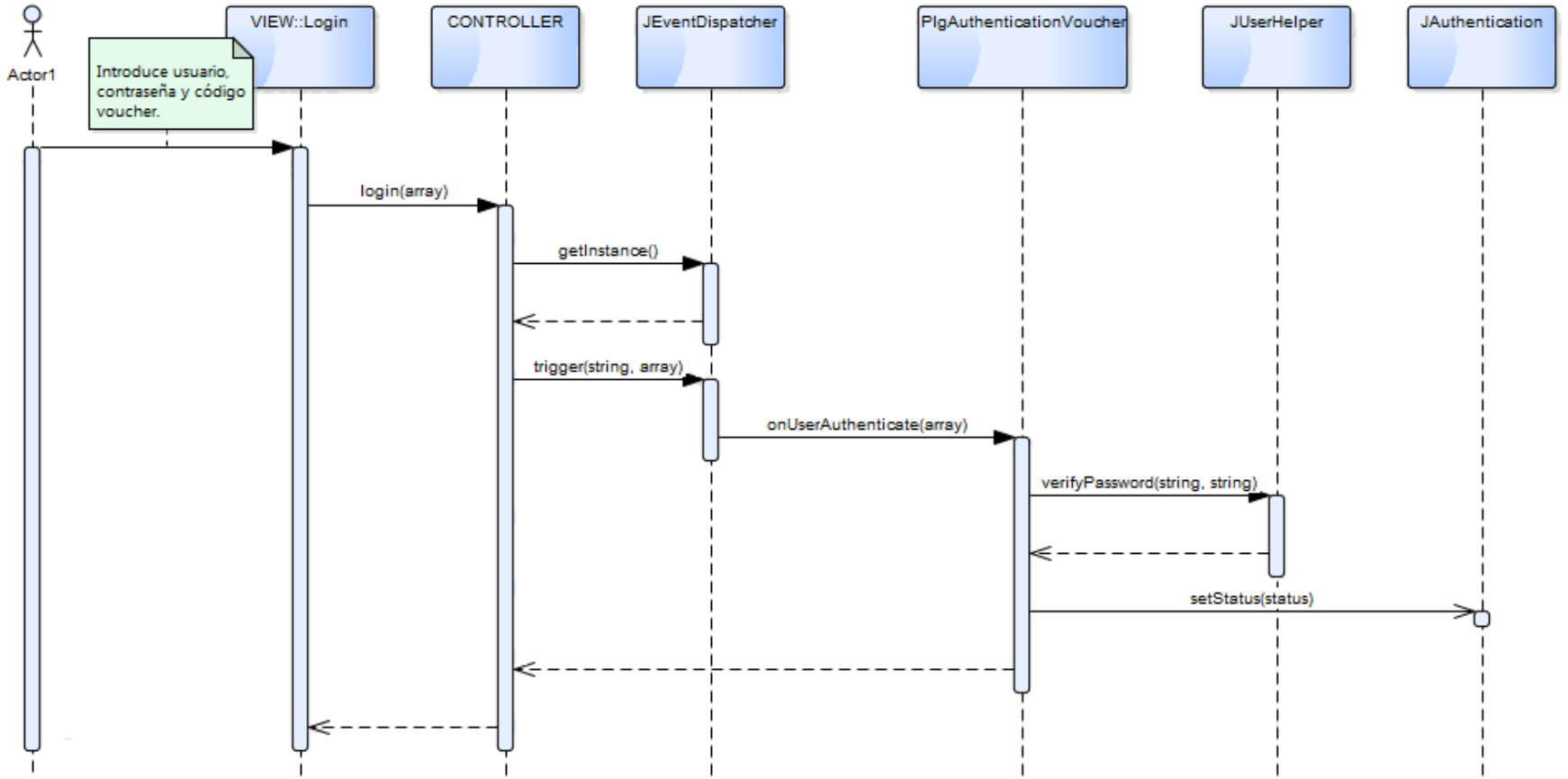

Figura 18. Diagrama de secuencia UC01. Recuperar acceso al sistema

4.3.2 Diagramas de secuencia del servidor de códigos *voucher*

En esta sección se presentan los diagramas de secuencia asociados a los casos de uso definidos en las secciones 3.5.3 y 3.5.4 de este documento.

UC02 – Generar código voucher

La interacción entre las clases necesarias para la generación de un código voucher, se presenta en la Figura 19 y en la Figura 20.

En el primer diagrama se representan las interacciones necesarias para cargar la pantalla desde la que se generará el código *voucher*. Cuando el usuario selecciona la opción de "Generar código *voucher*" través de un elemento de la interfaz de usuario, como por ejemplo un botón o un link, la vista le indica al controlador que debe recuperar la lista de solicitantes, es decir aquellos usuarios que tienen al usuario como asignado como ayudante.

El controlador invoca el método *listAskers* de la clase del modelo, *UserServiceImpl*, y éste a su vez lanza la petición a la clase *UserDAOImpl* la cual, mediante una consulta a la base de datos, recupera la información solicitada. Una vez que el controlador tiene la lista de posibles solicitantes, se la pasa a la clase de Spring *Model*, poniéndola a disposición de la vista, la cual se actualiza para mostrar dicha información.

Una vez cargada la pantalla, en el segundo diagrama se presentan las iteraciones que se llevan a cabo para la generación del código *voucher*. Primeramente, el usuario selecciona de la lista al solicitante y acto seguido pulsa la opción de generar código. La vista informa al controlador de la acción iniciada por el usuario, indicándole el identificador del solicitante seleccionado. En primer lugar se comprueba si existe alguna sesión de *vouching* activa asociada al usuario solicitante. De no ser así, el controlador invoca el método *getVoucher* de la clase *VoucherCodeGenerator*, obteniendo como resultado la generación del mismo.

Seguidamente, el controlador solicita a la clase *SessionServiceImpl* la creación de una nueva sesión de *vouching* en estado activo, asociada a los usuarios ayudante y solicitante y al código *voucher* generado.

Una vez realizada esta operativa, se actualiza la vista mostrando al usuario el código *voucher* generado.

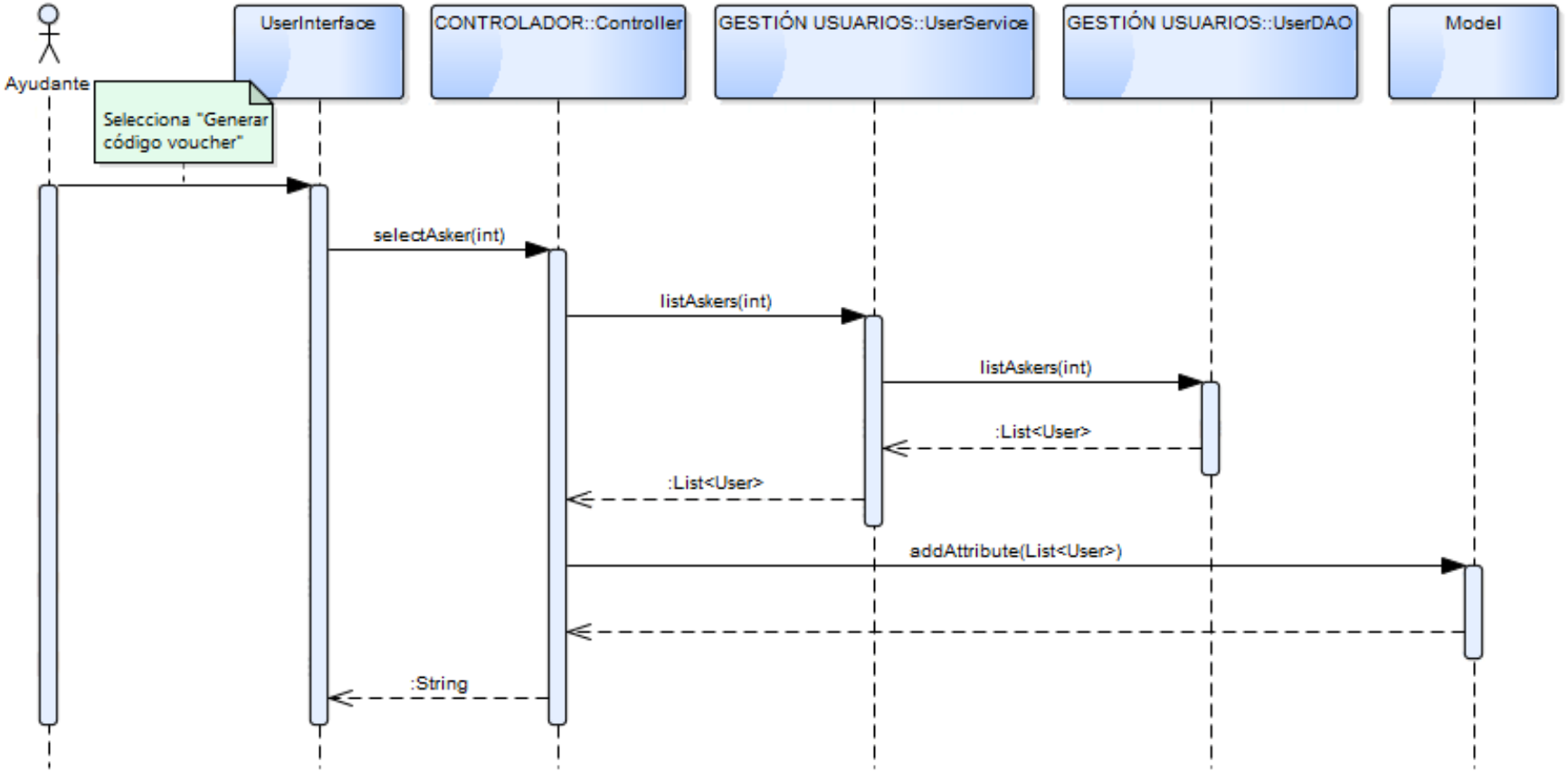

Figura 19. Diagrama de secuencia UC02. Generar código voucher (I)

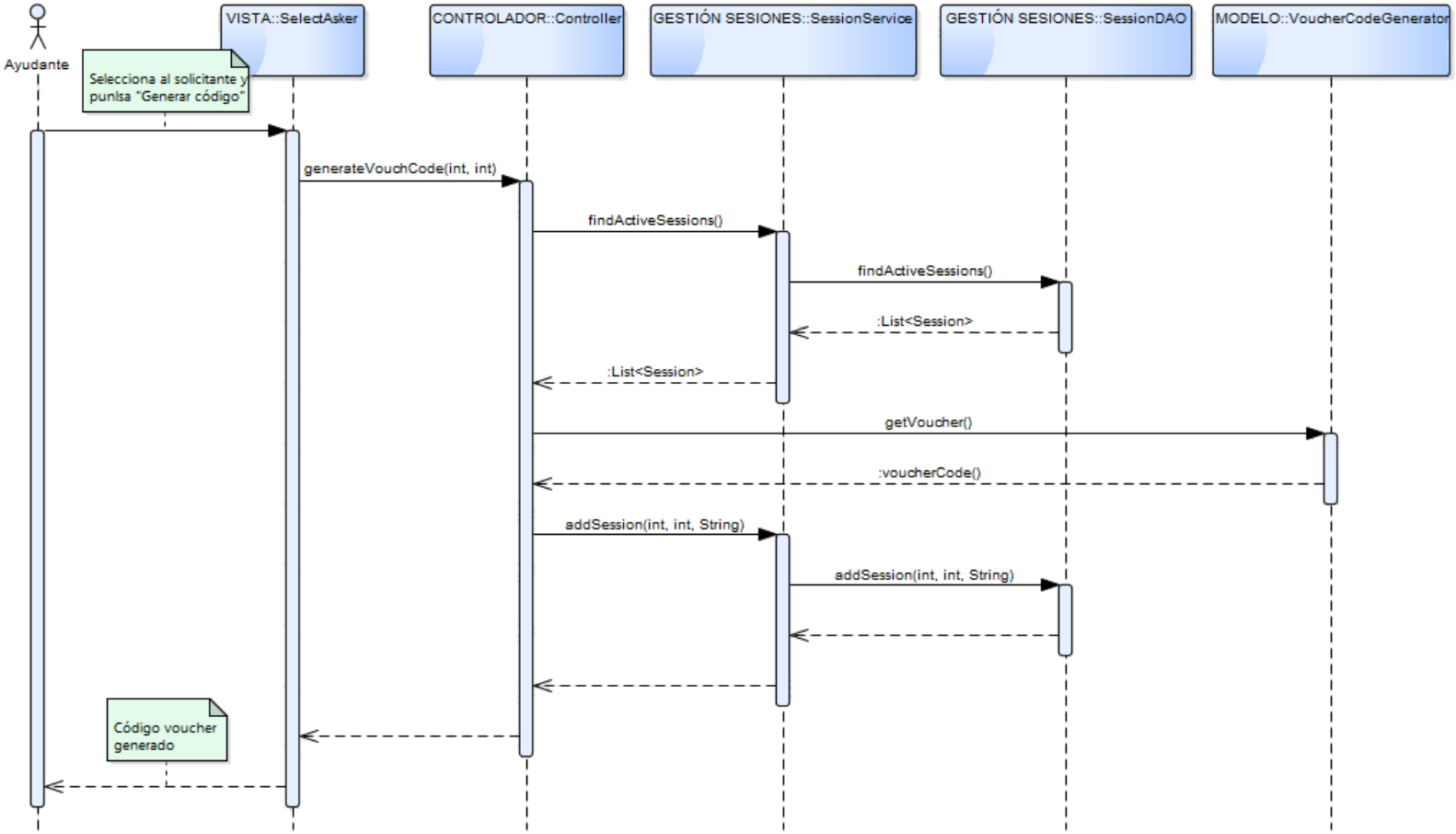

Figura 20. Diagrama de secuencia UC02. Generar código voucher (II)

UC03 – Consultar ayudante

La interacción entre las clases necesarias para dar de alta a un nuevo usuario se presenta en la Figura 21. Diagrama de secuencia UC03. Consultar ayudante Figura 21.

El usuario pulsa sobre la opción de consultar ayudante de la interfaz de usuario. La vista, a través de la llamada al método *helperDetails*, informa al controlador de la acción iniciada por el usuario. A su vez, el controlador solicita al servicio proporcionado modelo, *UserServiceImpl*, la lista de usuarios asignados como ayudantes. La clase *UserServiceImpl*, requiere que la clase encargada de realizar el acceso a los datos, *UserDAOImpl* recupere la lista.

Conviene resaltar, que pese a que en el párrafo anterior se hace referencia a la lista de ayudantes, en la actualidad el sistema está diseñado para soportar un único ayudante por usuario. La razón que ha motivado que el método *listUsersByAsker* se haya diseñado para retornar una lista, es la de mejorar la reutilización del mismo y facilitar la implementación del requisito de poder disponer de más de un ayudante por usuario en un futuro.

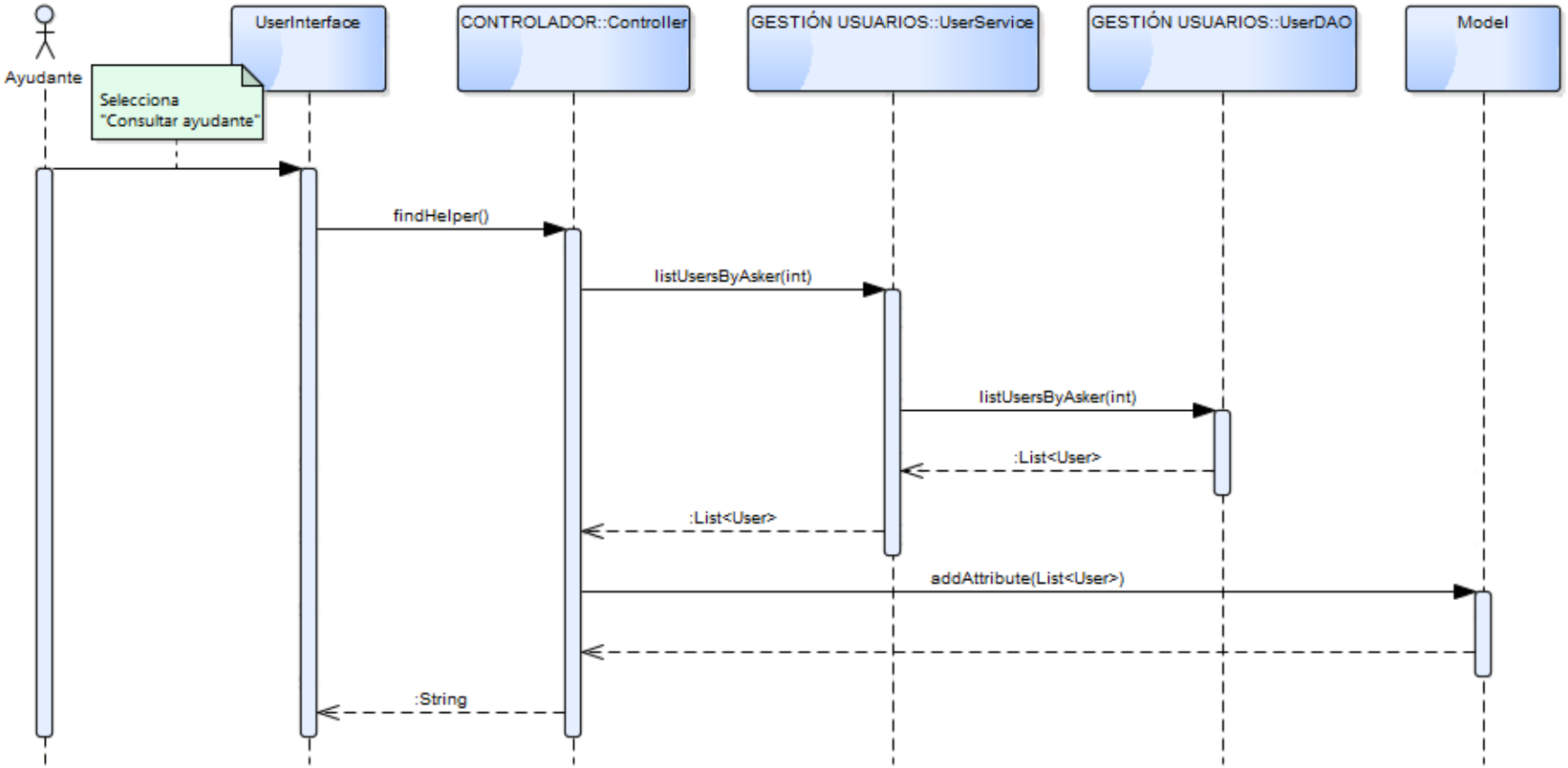

Figura 21. Diagrama de secuencia UC03. Consultar ayudante

UC04 – Crear usuario

La interacción entre las clases necesarias para dar de alta a un nuevo usuario se presenta en la Figura 22.

Primeramente, el administrador selecciona la opción correspondiente a crear usuario. La vista notifica de la acción a realizar al controlador y esté lleva a cabo las acciones necesarias para cargar el formulario de alta de usuario por pantalla. Para ello, en primera instancia es necesario recuperar la lista de usuarios del sistema, ordenados alfabéticamente, con el fin rellenar la lista desplegable del campo "Ayudante".

Una vez actualizada la vista con el formulario de alta, el usuario introduce los datos del nuevo usuario y selecciona la opción de crear. Previamente a almacenar el nuevo usuario, el controlador solicita al modelo, mediante el método *findByUserName*, que recupere si hay algún usuario almacenado que tenga asignado como identificador el introducido en el formulario. En caso de que la función retorne un valor nulo, se procede dar de alta el usuario. Para ello, el controlador, solicita al modelo que cree al nuevo usuario a través del método *addUser*.

Una vez creado, el controlador solicita a la clase *GoogleAuthenticator* la creación de la clave secreta que servirá al usuario para configurar en su dispositivo móvil la aplicación generadora de claves de *GoogleAuthenticator* con el fin de que pueda acceder al servidor de códigos *voucher*.

Por último, se envía un correo electrónico de bienvenida al usuario dado de alta. El controlador solicita esta acción a la clase *WelcomeEmailService*, pasándole los datos necesarios para componer la plantilla del mensaje.

Por último, la vista se actualiza mostrando la lista de usuarios del sistema

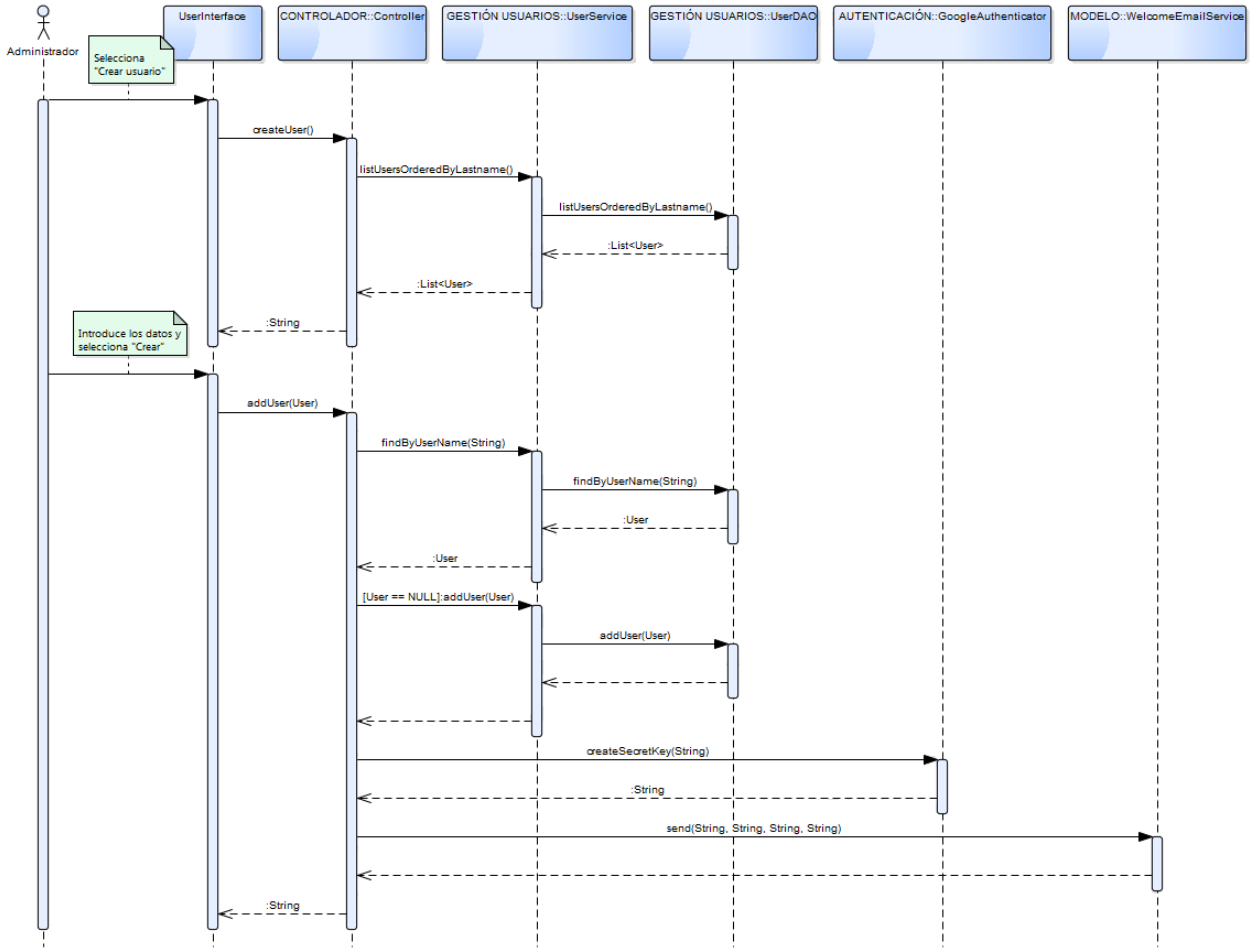

Figura 22. Diagrama de secuencia UC04. Crear usuario

UC05 – Consultar usuarios

La interacción entre las clases necesarias a la hora de consultar los usuarios del sistema se presenta en la Figura 23.

El administrador selecciona la opción correspondiente a gestionar usuarios. La vista informa de ello al controlador, el cual solicita al modelo que le facilite la lista, ordenada alfabéticamente, de usuarios dados de alta en el sistema. Por último, el controlador pone a disposición de la vista la lista obtenida mediante la llamada al método *addAttribute* de la clase de Spring *Model*.

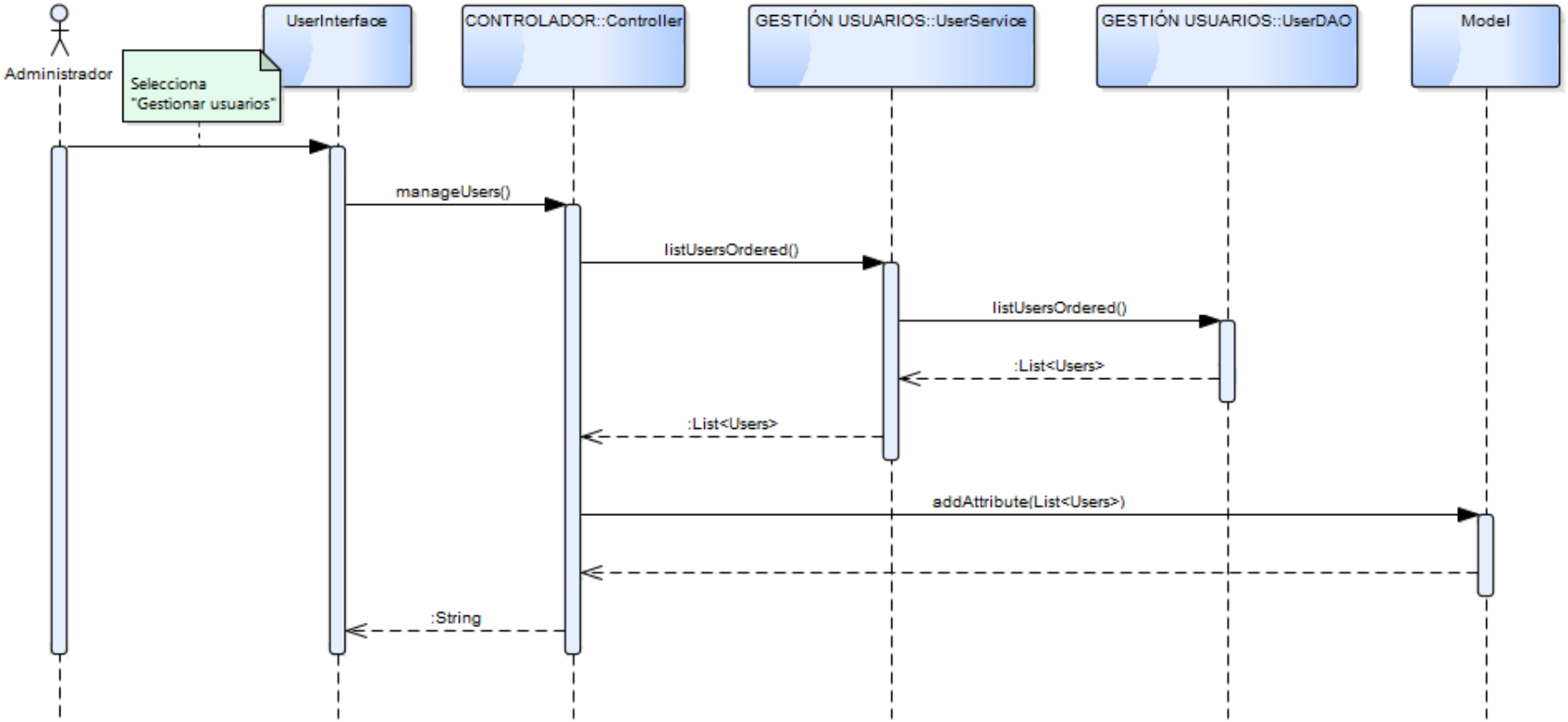

Figura 23. Diagrama de secuencia UC05. Consultar usuarios

UC06 – Actualizar usuario

La interacción entre las clases necesarias para poder llevar a cabo la actualización de los datos de usuario se presenta en la Figura 24.

Partiendo de la lista de usuarios generada en el caso de uso UC05, consultar usuarios, el administrador selecciona a un usuario de la lista y pulsa "Editar". El controlador realiza las tareas necesarias para cargar en pantalla el formulario de actualización.

El primer lugar, el controlador solicita al modelo la información del usuario que se desea actualizar, así como la lista de usuarios del sistema ordenada para rellenar el campo "Ayudante". Una vez recuperado la vista se actualiza mostrando al usuario el formulario con la información de usuario a modificar.

El usuario modifica los datos deseados y pulsa sobre la opción actualizar. El controlador es notificado mediante el método *updateUser*, y éste a su vez notifica al modelo para que actualice los datos del usuario.

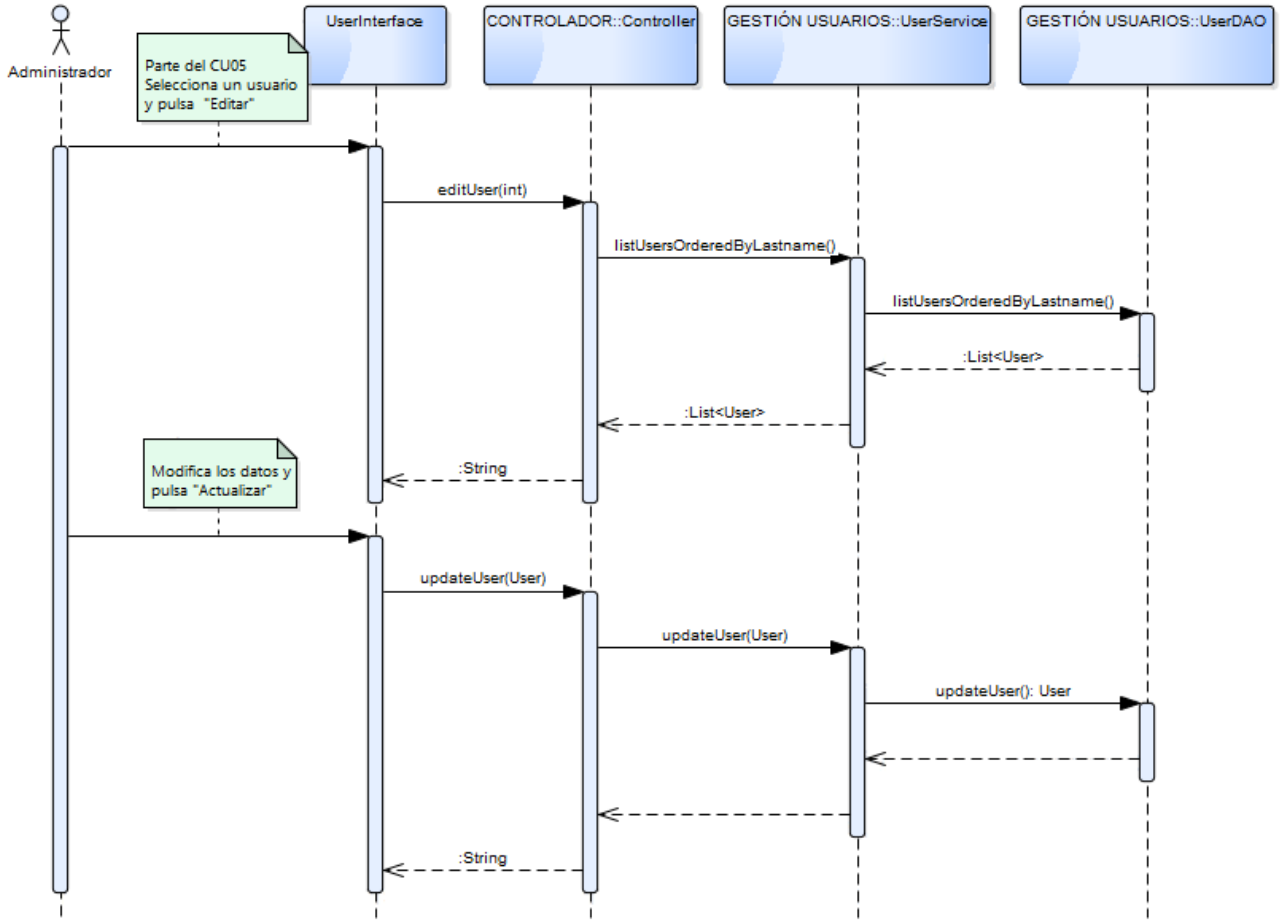

Figura 24. Diagrama de secuencia UC06. Actualizar usuario

UC07 – Eliminar usuario

La interacción entre las clases necesarias para eliminar a un usuario del sistema se presenta en la Figura 25.

Partiendo de la lista de usuarios generada en el caso de uso UC05, consultar usuarios, el administrador selecciona a un usuario de la lista y pulsa la opción de eliminar. La vista informa al controlador de la acción a llevar cabo mediante una llamada al método *removeUser*.

El controlador solicita al modelo la lista de usuarios que tienen asignado a dicho usuario como ayudante. En caso de que no haya ninguno, el controlador invoca el método *deleteUser* del modelo. En consecuencia el usuario es eliminado del sistema y la vista se actualiza mostrando la lista de usuarios restantes.

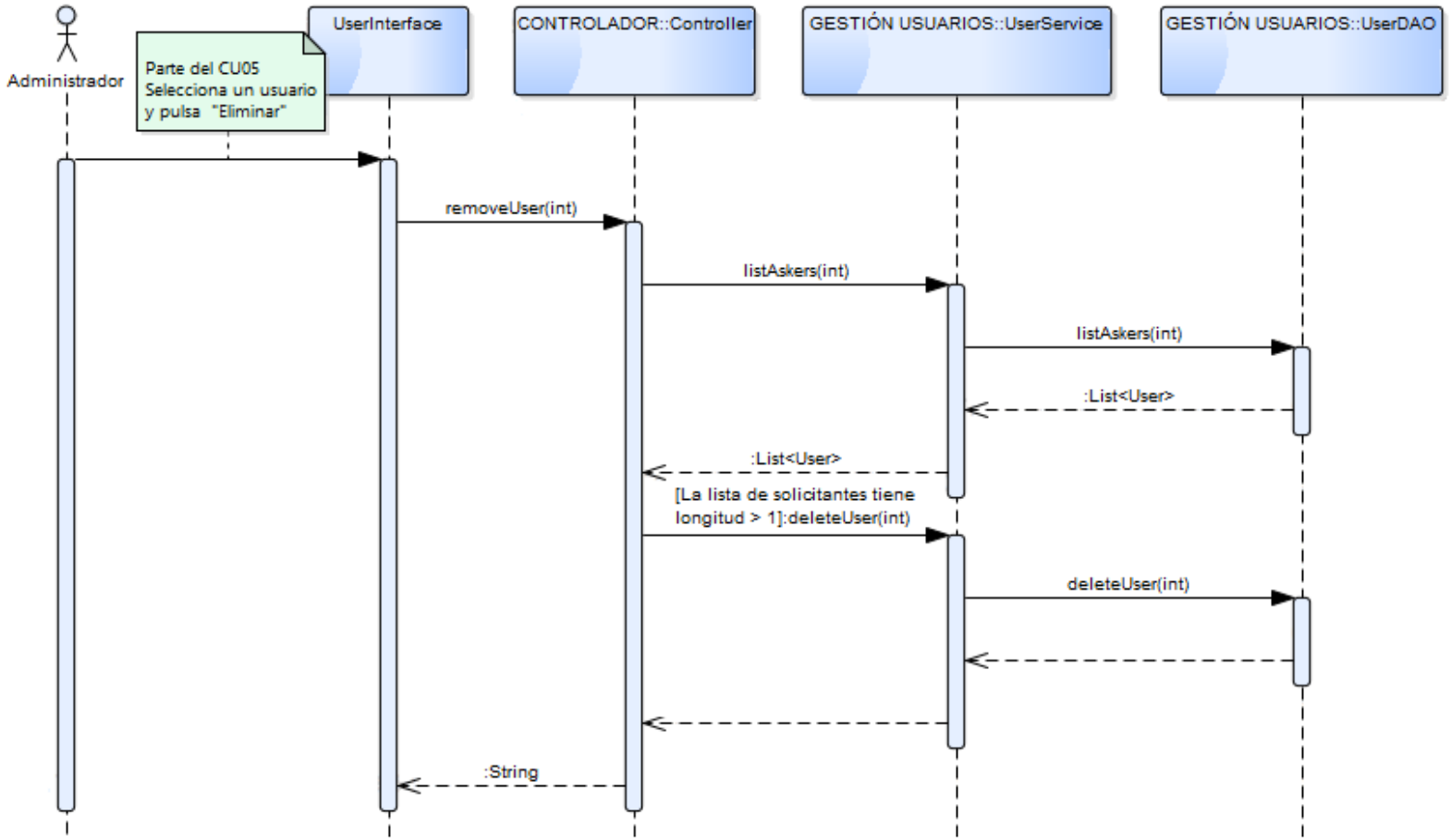

Figura 25. Diagrama de secuencia UC07. Eliminar usuario

UC08 – Consultar sesiones

La interacción entre las clases necesarias a la hora de consultar las sesiones activas en el sistema se presenta en la Figura 26.

El administrador selecciona la opción correspondiente a gestionar sesiones. La vista informa de ello al controlador, el cual solicita al modelo que le facilite la lista de sesiones en estado activo existentes en el sistema. Por último, el controlador pone a disposición de la vista la lista obtenida mediante la llamada al método *addAttribute* de la clase de Spring *Model*.

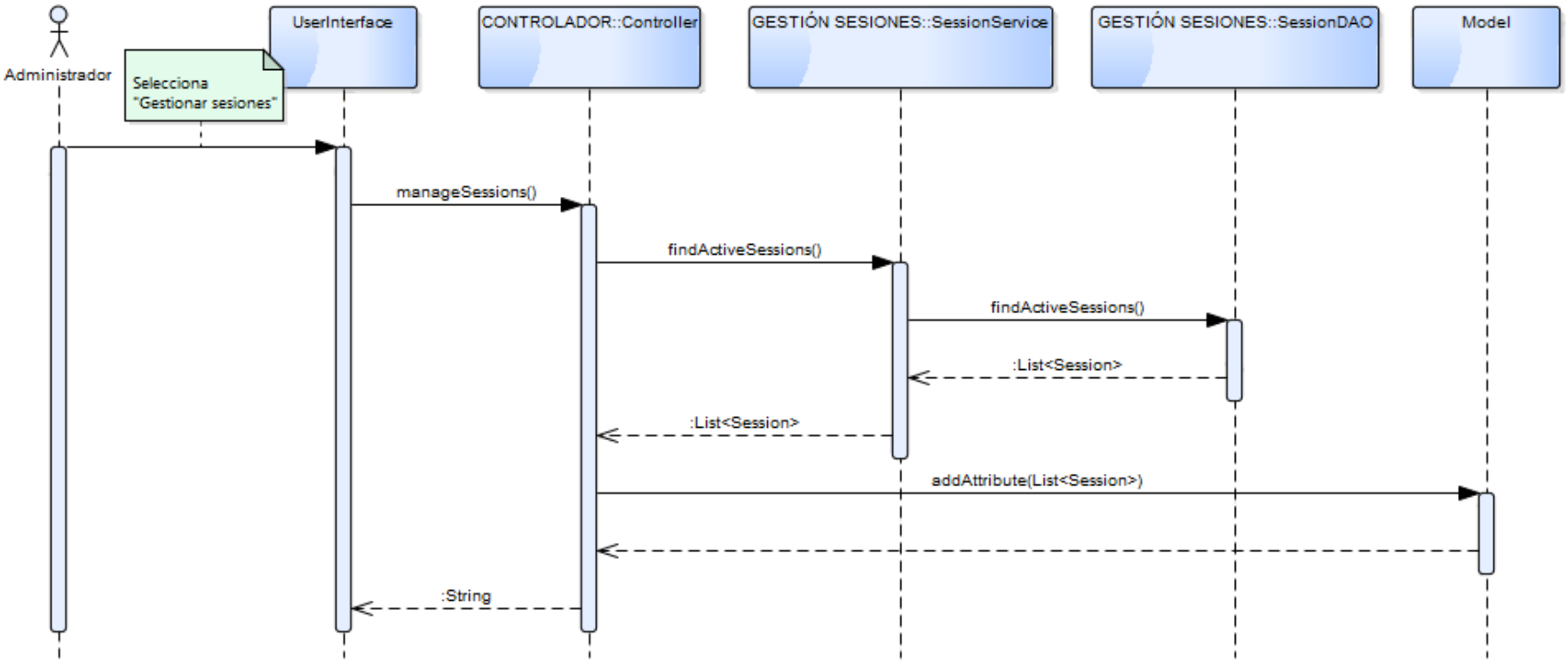

Figura 26. Diagrama de secuencia UC08. Consultar sesiones

UC09 – Inhabilitar sesión

La interacción entre las clases necesarias a la hora de consultar los usuarios del sistema se presenta en la Figura 27.

Partiendo de la lista de sesiones activas obtenida en el caso de uso UC08, consultar sesiones, el administrador selecciona a una sesión activa de la lista y pulsa la opción de inhabilitar. La vista informa al controlador de la acción a llevar cabo mediante una llamada al método *inactivateSession*.

El controlador a su vez solicita al modelo que actualícela sesión, para ello invoca a la función *updateSession* pasándole como atributo la sesión con el estado a inactivo.

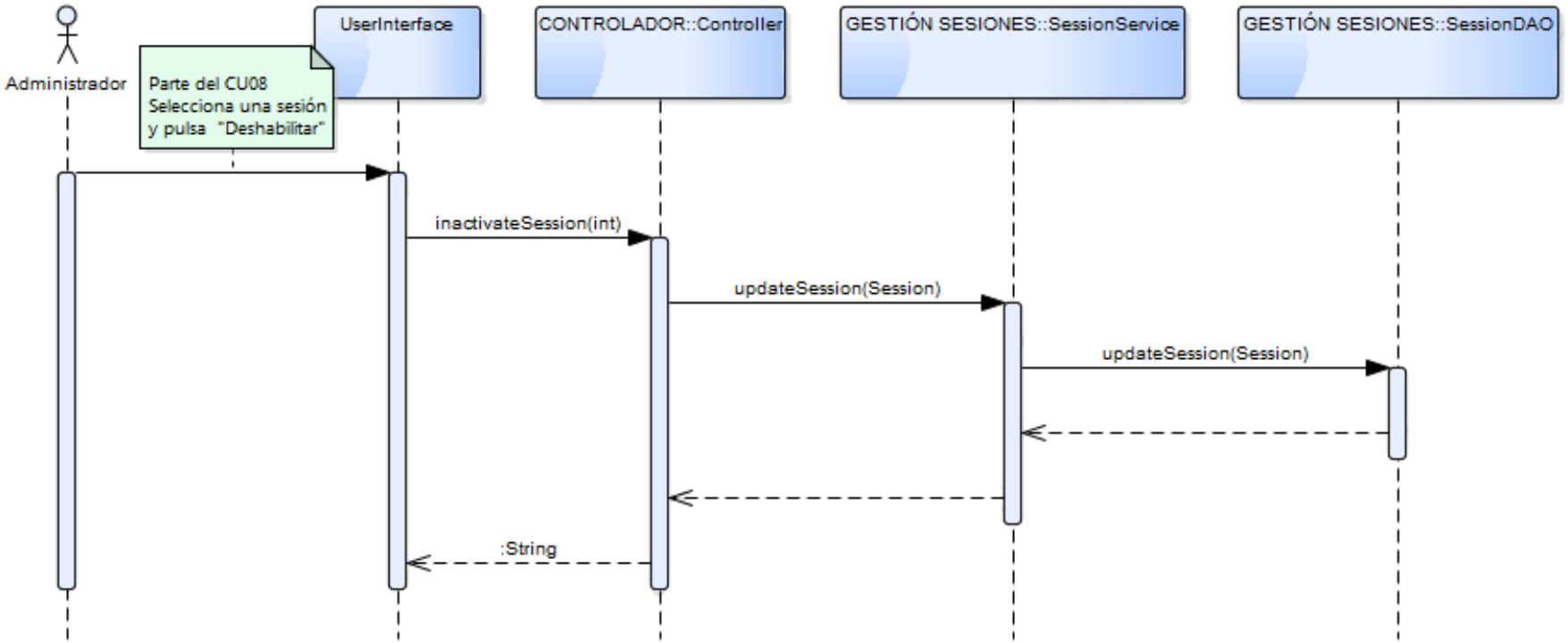

Figura 27. Diagrama de secuencia UC09. Inhabilitar sesión

Capítulo 5

Implementación del software

Una vez realizados el análisis y el diseño, el presente capítulo se centra en la fase de implementación del software. En las siguientes secciones se describirán los aspectos más relevantes del proceso de implementación de ambas aplicaciones. La sección 5.1 detalla las decisiones más importantes que se han tomado durante la fase de implementación del software. En la sección 5.2 se incluye el resultado tras la ejecución del plan de pruebas definido en la fase de análisis.

5.1 Decisiones de implementación

Tras haber llevado a cabo un exhaustivo análisis y el posterior diseño, en esta sección se describen las decisiones más relevantes relativas a la implementación del software que han contribuido a la consecución con éxito de esta fase del proyecto.

5.1.1 Gestión de dependencias

Cuando se trabaja con Java/JavaEE en un proyecto relativamente complejo, es común tener que manejar un número elevado de librerías. Pero en general no basta con saber qué librería se necesita, sino que también es necesario conocer la versión exacta. A su vez, es frecuente que unas librerías dependan de otras para su correcto funcionamiento.

Para solventar este problema y facilitar así la implementación, se ha decidido utilizar Maven (v.4.0.0) como repositorio donde almacenar y mantener las dependencias software del proyecto. Maven es una herramienta de software orientada a la creación y gestión de proyectos Java desarrollada por Apache, que puede integrarse con diversos IDE, entre los que destaca Eclipse (24).

La unidad central de todo proyecto Maven es el POM (*Project Object Model*). Es un fichero XML que contiene información y detalles de configuración acerca del proyecto. En él se recogen además, todas las dependencias externas.

5.1.2 Implementación de la autenticación de doble factor

Es posible implementar un sistema de autenticación de doble factor basado en usuario y contraseña y clave OTP de distintas formas. Por un lado puede ser implementarlo en dos pasos: primeramente el usuario es instado a introducir su usuario y contraseña, y en caso de ser ambos satisfactorios el usuario quedaría pre-autenticado en el sistema (por ejemplo se le asignaría el rol PRE_AUTH_USER). A continuación se le solicita la clave OTP y una vez validada se le asignarían los permisos necesarios de acceso al sistema (rol ROLE_USER). Otra forma de implementar la autenticación de doble factor es solicitando

al usuario su nombre de usuario, contraseña y clave OTP en un único formulario, autenticando y autorizando al usuario en el sistema en un único paso.

Por razones de usabilidad, y para mejorar la eficiencia del método de autenticación, se ha decidido implementar la autenticación de doble factor del servidor de códigos *voucher* en un único paso.

Spring Security proporciona de forma nativa la autenticación a través de formulario para usuario y contraseña. Sin embargo, para la implementación del mecanismo de doble factor (usuario, contraseña y clave OTP) ha sido necesario realizar un desarrollo a medida, extendiendo algunas de las clases implicadas en el flujo de autenticación/autorización de Spring Security y que se detallan a continuación:

- **TwoFactorAuthenticationFilter:** extiende de la clase abstracta *AbstractAuthenticationProcessingFilter* proporcionada por Spring Security. Los filtros son los encargados de obtener el nombre de usuario y la contraseña introducidos en el formulario de *login* y convertirlos en un objeto *UsernamePasswordAuthenticationToken* que será enviado al *AuthenticationManager*. Esta clase ha sido extendida para que además de los dos campos ya mencionados, obtenga también la clave OTP del formulario de *login* del servidor de códigos voucher y lo transforme en un objeto *TwoFactorAuthenticationToken*.

- **TwoFactorAuthenticationToken:** extiende la funcionalidad de la clase *UsernamePasswordAuthenticationToken*, cuya finalidad es agrupar en un único objeto las credenciales usuario y contraseña, para incorporar además la clave OTP.

- No ha sido necesario realizar ninguna modificación sobre el *authentication manager*. Su objetivo consiste en identificar al *authentication provider* capaz de manejar, en este caso, *TwoFactorAuthenticationTokens*, que será la clase *TwoFactorAuthenticationProvider*, a la cual le pasa el *token* para que se lleve cabo la autenticación.

- **TwoFactorAuthenticationProvider:** extiende de la clase abstracta *AbstractUserDetailsAuthenticationProvider* para permitir manejar la

clave OTP además del usuario y la contraseña. El *authentication provider* es el encargado de llevar a cabo el proceso de autenticación propiamente dicho. Primeramente se invoca al método de la interfaz *loadUserByUsername*, el cual devuelve un objeto *UserDetails* con los información del usuario (usuario, contraseña, clave OTP y rol/roles) o una excepción *UsernameNotFound* si no ha encontrado al usuario en la base de datos. A continuación se comparan las contraseñas almacenadas en el objeto *TwoFactorAuthenticationToken* y en el objeto *UserDetails* (contiene el hash de la contraseña almacenado en la base de datos). En el caso de coincidir tanto las contraseñas como la clave OTP el usuario queda automáticamente autenticado en el sistema y se pasa el control al AccessDecisionManager encargado de realizar la autorización.

5.1.3 Plantillas de los correos electrónicos

Para la implementación de los correos electrónicos automáticos enviados por el servidor de códigos *voucher*, se ha decidido utilizar el gestor de plantillas *Velocity* de Apache.

Velocity permite generar plantillas con contenido dinámico en formato HTML e implementa el patrón MVC (25). Las plantillas de los correos electrónicos se almacenan en ficheros de texto, evitando así embeber el texto del correo dentro del código Java de la aplicación. Esta característica facilita en gran medida, no sólo la implementación, si no el mantenimiento y la legibilidad del código.

5.2 Resultado de las pruebas de aceptación

En esta sección se muestran los resultados de las pruebas de aceptación definidas en la sección 3.7. Como se puede apreciar en la tabla, tras su ejecución, todos los casos de prueba han sido superados con éxito:

Servidor de códigos *voucher*

Pruebas de aceptación		
ID	**Requisitos probados**	**Resultado**
PA-01	RF01	Superada
PA-02	RF02	Superada
PA-03	RF03, RF-05	Superada
PA-04	RF03, RI-04	Superada
PA-05	RF04	Superada
PA-06	RF-06, RF-09	Superada
PA-07	RF-06, RI-01	Superada
PA-08	RF-10	Superada
PA-09	RF-07, RI-02	Superada
PA-10	RF-08	Superada
PA-11	RF-08, RI-03	Superada
PA-12	RF-11	Superada
PA-13	RF-12	Superada
PA-14	RF-13	Superada
PA-15	RF-14	Superada
PA-16	RF-15	Superada

Tabla 17. Resultados de las pruebas de aceptación del servidor de códigos voucher

Servidor de códigos *voucher*

Pruebas de aceptación		
ID	**Requisitos probados**	**Resultado**
PA-17	RF-16, RF-17	Superada
PA-18	RF-16, RF-17, RI-05	Superada
PA-19	RF-16, RF-17, RI-06	Superada

Tabla 18. Resultados de las pruebas de aceptación del componente de autenticación de emergencia

Capítulo 6

Conclusión y líneas futuras

Tras la finalización de las distintas fases de desarrollo del proyecto, en la sección 6.1 se detallan las conclusiones obtenidas a lo largo de este proceso. Asimismo, en el apartado 6.2 se sugieren una serie de líneas futuras de investigación susceptibles de dar continuidad al trabajo desarrollado en el presente proyecto.

6.1 Conclusiones

A continuación se describen las conclusiones más relevantes obtenidas tras el desarrollo de este proyecto.

6.1.1 Aportaciones

En el presente trabajo se desarrolla un mecanismo de autenticación de emergencia para la plataforma Joomla! basado en el concepto de *vouching* y en las relaciones sociales. Por tanto, como principal aportación, cabe destacar la seguridad adicional que proporciona al mecanismo de autenticación de doble factor de Joomla! presente desde su versión 3.2.

Este trabajo explora el vínculo entre la autenticación y las relaciones sociales, algo que en la vida real se realiza de forma automática en el día a día de las personas, pero que llevado al ámbito de los sistemas informáticos es un área sobre el que se ha profundizado muy poco, dado que la gran mayoría de los prototipos existentes están aún en fase experimental. Asimismo, este proyecto puede ser un punto de partida de cara a continuar las investigaciones en esta línea, para su posterior implementación en sistemas finales.

Con este trabajo no sólo se demuestra que este tipo de autenticación es viable, sino que puede aportar grandes beneficios. Por ejemplo, aprovechar las relaciones interpersonales que se establecen entre los empleados, puede ser una alternativa a los costosos mecanismos de recuperación de cuentas actuales en las empresas, tales como los basados en equipos de soporte y *Service Desks*.

6.1.2 Dificultades del proyecto

La realización de un trabajo de estas dimensiones, que implica la creación de un sistema pasando por todas las fases del desarrollo, es un proceso laborioso que supone un gran esfuerzo y dedicación. Es casi imposible, que ante un proyecto de esta magnitud, uno no se tope con ciertas dificultades y obstáculos que haya que sortear.

Si bien es cierto que durante la realización de este proyecto no se han encontrado dificultades insalvables, si cabe destacar el esfuerzo que conllevó el estudio y análisis de

la situación actual, que fueron necesarios antes de comenzar con las etapas propias del ciclo de desarrollo del software.

Por otro lado, al tratarse de un proyecto enmarcado en el ámbito de la seguridad informática, ha supuesto un esfuerzo adicional la gran cantidad de aspectos a tener en cuenta con el fin de desarrollar una aplicación web lo suficientemente robusta y segura, tales como el almacenamiento del *hash* de las contraseñas o la protección frente a ataques CSRF (del inglés *Cross Site Request Forgery*) entre otros.

Otro obstáculo que vale la pena resaltar fue la necesidad de tener que implementar desde cero la RFC 6238 correspondiente al algoritmo TOTP, con el objetivo de poder implementar la autenticación de doble factor con *Google Authenticator* para el servidor de códigos *voucher*.

6.1.3 Conclusiones personales

Una de las mayores dificultades experimentadas a nivel personal durante la realización de este Proyecto de Fin de Carrera ha sido compaginarlo con un trabajo a jornada completa. Debido a los altibajos en cuanto a carga de trabajo, en ocasiones ha sido difícil dedicarle al proyecto el tiempo que requiere, impactando inevitablemente en la planificación original. Sin embargo, esta situación ha conllevado a que cada hito completado me haya reportado una gran satisfacción y confianza en el esfuerzo realizado.

Otra dificultad a nivel personal ha sido la necesidad de desarrollar utilizado tecnologías con las que no estaba familiarizada hasta este momento. Como por ejemplo el *framework Spring MVC* y *Spring Security*, o el desarrollo de extensiones para Joomla! El aprendizaje de estas tecnologías ha supuesto un avance inicial más lento, así como un mayor esfuerzo, pero sin duda ha contribuido de forma positiva a la calidad final del software.

A pesar lo anterior, la realización de este proyecto ha sido un proceso gratificante. Me ha permitido profundizar en un área que me gusta, el de la autenticación, el cual ya había tenido la oportunidad de explorar gracias a mi trabajo como consultor de seguridad. Ha sido verdaderamente interesante investigar hacia nuevas líneas de autenticación poco exploradas que permitan ampliar los tres factores ya conocidos.

6.2 Líneas futuras

En esta sección se detallan una serie de posibles mejoras a la solución propuesta, que dejan una puerta abierta a la investigación para futuros proyectos.

6.2.1 Incrementar el número de ayudantes por usuario

El sistema de *vouching* desarrollado en este proyecto está diseñado de tal forma que un usuario sólo puede tener un único ayudante asignado. En el mundo real, sería deseable que un sistema de estas características soportara más de un ayudante por usuario, de tal forma que si un ayudante no está disponible o no tiene acceso a un dispositivo conectado a Internet, el solicitante pueda contactar con otro ayudante, aumentando así las probabilidades de recuperar el acceso a su cuenta.

El número óptimo de ayudantes que puede tener un usuario es un dato que debe ser analizado cuidadosamente. Un número muy elevado de ayudantes puede debilitar el sistema haciéndolo más inseguro. Esto es debido a que cuanto mayor es el número de ayudantes, mayor es también el número de puntos de ataque, ya que pueden ser engañados por un impostor que se hace pasar por el solicitante. Sin embrago, un número demasiado bajo de ayudantes, puede reducir significativamente la eficacia del sistema como método de autenticación de emergencia.

6.2.2 Contraseña temporal

Cabe esperar que un usuario que no dispone de su método de autenticación primario, necesite acceder al portal de Joomla! en más de una ocasión antes de recuperarlo. Por ello, es posible configurar el tiempo de validez de la sesión de *vouching*, permitiendo al usuario utilizar su código *voucher* para acceder al portal en más de una ocasión hasta que expire la sesión.

Sin embargo, el código *voucher* es una cadena de caracteres alfanuméricos aleatorios y que puede tener una longitud variable, lo cual lo convierte en un código difícil de recordar para el usuario. Ante esta situación, los usuarios tienden a emplear malas prácticas para recordar los códigos, como apuntarlo en papel o en un archivo en el propio

ordenador y tal y como se ha comentado en capítulos anteriores este comportamiento compromete la seguridad de todo el sistema.

Para evitar esto, una posible mejora sería que una vez validado el código *voucher* introducido por el usuario, éste sea instado a introducir una contraseña temporal que sea capaz de recordar. Esta contraseña podría ser utilizada por el usuario para autenticarse en el portal de Joomla! tantas veces sea necesario durante el período de validez de la sesión de *vouching*, quedando el código *voucher* inhabilitado después de su uso.

6.2.3 Aceptación de la relación ayudante-solicitante

Una mejora al proceso de inicialización, cuando se da de alta un usuario en el servidor de códigos voucher y se le asigna a uno o varios ayudantes, sería la aceptación por parte de las partes implicadas, ayudante y solicitante, de que ambos usuarios se conocen y están en pleno conocimiento de sus roles.

Este proceso de aceptación de la relación ayudante-solicitante podría llevarse a cabo de distintas formas, las cuales sería preciso analizar. Bien podría realizarse a través de un correo electrónico enviado a cada una de las partes con un enlace para que acepten la relación o bien mediante un canal independiente (*Out of Band*) como puede ser un mensaje de texto o una llamada telefónica automatizada.

6.2.4 Registro de eventos

Es altamente deseable que todo sistema que incluye control de accesos registre toda su actividad en ficheros de *log*. De esta manera, no sólo es posible detectar accesos no autorizados al sistema, sino que además puede rastrearse toda la actividad de los usuarios que tienen sesiones de *vouching* activas, pudiendo identificar en qué casos se hace un uso indebido del sistema.

Por tanto, se propone como mejora la implementación de sistema de registro de eventos que permita a un usuario con permisos elevados, como puede ser el administrador, auditar en tiempo real la actividad llevada a cabo por los usuarios.

Existen infinidad de *frameworks* de *logging* para Java disponibles con licencia *OpenSource*. Entre los más conocidos se encuentra Log4j 2, el cual ofrece mejoras significativas en comparación con su predecesor Log4j.

6.2.5 Accesibilidad

Otra línea de continuidad para el presente proyecto es el de convertir tanto el servidor de códigos voucher como el componente de autenticación de emergencia, en aplicaciones más accesibles para todos los usuarios.

Por un lado es posible traducir la interfaz de usuario a más idiomas (a parte del inglés y el español) pudiendo llegar a un mayor número de usuarios. Asimismo, es posible adaptar el diseño de la aplicación de forma que personas con algún tipo de discapacidad (visual, auditiva, cognitiva, etc.) puedan interactuar al igual que otro usuario cualquiera. Para ello pueden seguirse las especiaciones en materia de accesibilidad del W3C.

Glosario

API	*Application Programming Interface*
ASP	*Active Server Pages*
BYOD	*Bring Your Own Device*
CA	*Certification Authority*
CGI	*Common Gateway Interface*
CSRF	*Cross-Site Request Forgery*
DAO	*Data Access Object*
ESA	*European Space Agency*
GPS	*Global Positioning System*
HTML	*HyperText Markup Language*
JEE	*Java Enterprise Edition*
JSP	*Java Server Pages*
MAC	*Media Access Control*
MIME	*Multipurpose Internet Mail Extension*
MITM	*Man In The Middle*
MVC	*Model View Controller*
NIST	*National Institute of Standards and Technology*
OOB	*Out Of Band*
ORM	*Object Relational Mapping*
OTP	*One Time Password*
PKI	*Public Key Infrastructure*
POM	*Project Object Model*
TOTP	*Time-based One Time Password*
USB	*Universal Serial Bus*
W3C	*World Wide Web Consortium*
XML	*Extensible Markup Language*

Referencias

1. **J. Brainard, A. Juels, R.L. Rivest, M. Szydlo, and M. Yung.** *Fourth-Factor Authentication: Somebody You Know.* Alexandria,Virginia, USA : ACM, 2006. 1595935185/.
2. **wikibooks.** *Fundamentals of Information Systems Security/Access Control Systems.*
3. **T. Henry.** *Fundamentals of Authentication.* s.l. : Gartner Inc., 2013. G00238761 .
4. **Zage, S. Choi and D.** *Addressing Insider Threat using "Where You Are" as Fourth Factor Authentication.* Albuquerque, NM, USA : Sandia National Laboratories. 87185-9300.
5. *Improving password security and memorability to protect personal and organizational information.* **K. L. Vu, R. W. Proctor, A. Bhargav-Spantzel, B. Tai, J. Cook, and E. Schultz.** s.l. : International Journal of Human-Computer Studies , 2007, Vol. 65. 744–757.
6. **Gartner Research.** *Best Practices for Managing Passwords: Self-Service.* s.l. : Gartner Inc., 2003. TU-20-2040.
7. **RSA Security.** RSA security survey reveals multiple passwords creating security risks and end-user frustration. *www.rsasecurity.com.* [En línea] 2005. www.rsasecurity.com.
8. **Milletary, J.** *Technical Trends in Phishing Attacks.* s.l. : Carnegie Mellon University, 2005.
9. **F. Tari, A. Ant Ozok, and S. H. Holden.** *A Comparison of Perceived and Real Shoulder-surfing: Risks between Alphanumeric and Graphical Passwords.* Pittsburgh, PA, USA : s.n., 2006.
10. **Allan, A.** *Good Authentication Choices: Evaluating Phone-as-a-Token Authentication Methods.* s.l. : Gartner Inc., 2012. 234565.
11. **M. Tistarelli, and M. Nixon.** *Advances In Biometrics.* s.l. : Springer-Verlag Berlin Heidelberg, 2009. 03029743.
12. **NIST.** *Electronic Authentication Guideline.* Gaithersburg, MD : National Institute of Standards and Technology, 2013. 800-63-1.
13. **S. Schechter, S. Egelman and R.W. Reeder.** *It's Not What You Know, But Who You Know: A Social Approach to Last-Resort Authentication.* Boston, MA, USA : ACM Press, 2009.
14. **S. Schechter, and R.W. Reeder.** *When the Password Doesn't Work: Secondary Authentication for Websites.* s.l. : IEEE Computer and Reliability Societies, 2011.
15. **B. Soleymani, and M. Maheswaran.** *Social Authentication Protocol for Mobile Phones.* Montreal, QC, Canada : IEEE, 2009. 978-0-7695-3823-5/09.
16. Joomla! 3.2 Beta1 Released. [En línea] 2013 de October de 11. https://www.joomla.org/announcements/release-news/5511-joomla-3-2-beta1-released.html.
17. M. Dexter, and L. Landry. *Joomla programming.* s.l. : Pearson Education Inc., 2012. 978-0-13-278081-0.

18. **J. Walden, M. Doyle, R. Lenhof, and J. Murray.** *Java vs. PHP: Security Implications of Language Choice for Web Applications.* Kentucky : Northern Kentucky University, 2010.
19. **S. Trent, M. Tatsubori, T. Suzumura, A. Tozawa, and T. Onodera.** *Performance Comparison of PHP and JSP as Server-Side Scripting Languages.* Japan : IBM Tokyo Research Laboratory, 2008.
20. Guía ESA para la definición de requisitos. [En línea] http://cisas.unipd.it/didactics/STS_school/Software_development/Guide_to_the_SW_requirements_definition_phase-0503.pdf.
21. **E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides.** *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 2009. 0-201-63361-2.
22. Srping Web MVC framework. [En línea] http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html.
23. *RFC 6238 - TOTP: Time-Based One-Time Password Algorithm.* [En línea] https://tools.ietf.org/html/rfc6238.
24. Apache Maven Project. [En línea] http://maven.apache.org/.
25. Apache Velocity - Guía del usuario de Velocity. [En línea] http://velocity.apache.org/engine/devel/translations/user-guide_es.html.
26. Microsoft Windows. [En línea]
27. *Microsoft Office 2013.* [En línea] https://products.office.com/es-es/.
28. Microsoft Project 2013. [En línea] https://products.office.com/en-us/project/.
29. Symantec Endpoint Protection. [En línea] http://www.symantec.com/es/es/endpoint-protection/.
30. *Eclipse Kepler.* [En línea] http://eclipse.org/kepler/.
31. *jEdit Programmer's Text Editor.* [En línea] http://www.jedit.org/.
32. *Sparx Systems Enterprise Architect.* [En línea] http://www.sparxsystems.com/products/ea/.
33. *WAMP Server.* [En línea] http://www.wampserver.com/en/.

Anexo I:

Gestión del proyecto

1.1 Planificación del trabajo

En esta sección se detalla la planificación inicial y real del proyecto, así como un análisis de la desviación entre ambas planificaciones y la justificación.

Para el proceso de desarrollo se ha seguido el modelo en cascada, en el cual al finalizar cada una de las fases se ha llevado a cabo una revisión para determinar si se continúa a la siguiente fase o por el contrario es necesario realizar algún cambio en alguna de las fases anteriores.

1.1.1 Planificación inicial

A continuación se detalla la planificación inicial, divida en las fases que conforman el desarrollo. Para cada una de las fases se ha especificado el tiempo estimado para su completitud, expresado tanto en horas como en días.

La planificación inicial consta de 685 horas, repartidas en 155 días entre el 15/09/2014 y el 16/02/2015. Debido a la necesidad de compaginar un trabajo a jornada completa, con jornada partida de 09:00 a 19:00 horas, se han estimado 3 horas de trabajo en días laborables y 8 horas de trabajo al día durante los fines de semana, lo que conforma un total de 31 horas semanales.

Nombre	Horas	Duración	Fecha de inicio	Fecha de fin
Proyecto	**685**	**155 días**	**15/09/2014**	**16/02/2015**
Planificación	**9**	**3 días**	**15/09/2014**	**17/09/2014**
Estado del Arte	**90**	**20 días**	**18/09/2014**	**07/10/2014**
Análisis	**87**	**19 días**	**08/10/2014**	**26/10/2014**
Planteamiento del problema	3	1 día	08/10/2014	08/10/2014
Perspectiva de la solución	14	3 días	09/10/2014	11/10/2014
Estudio tecnológico	17	4 días	12/10/2014	15/10/2014
Definición de la arquitectura preliminar	14	3 días	16/10/2014	18/10/2014
Definición de casos de uso	14	3 días	19/10/2014	21/10/2014
Análisis de requisitos	17	4 días	22/10/2014	25/10/2014
Definición de casos de prueba	8	1 día	26/10/2014	26/10/2014
Diseño	**124**	**28 días**	**27/10/2014**	**23/11/2014**
Definición de la arquitectura definitiva	9	3 días	27/10/2014	29/10/2014
Diseño de clases	53	11 días	30/10/2014	09/11/2014
Diagramas de secuencia	62	14 días	10/11/2014	23/11/2014
Implementación	**341**	**77 días**	**24/11/2014**	**08/02/2015**
Pruebas	**34**	**8 días**	**09/02/2014**	**16/02/2014**

Tabla 19. Planificación inicial del proyecto

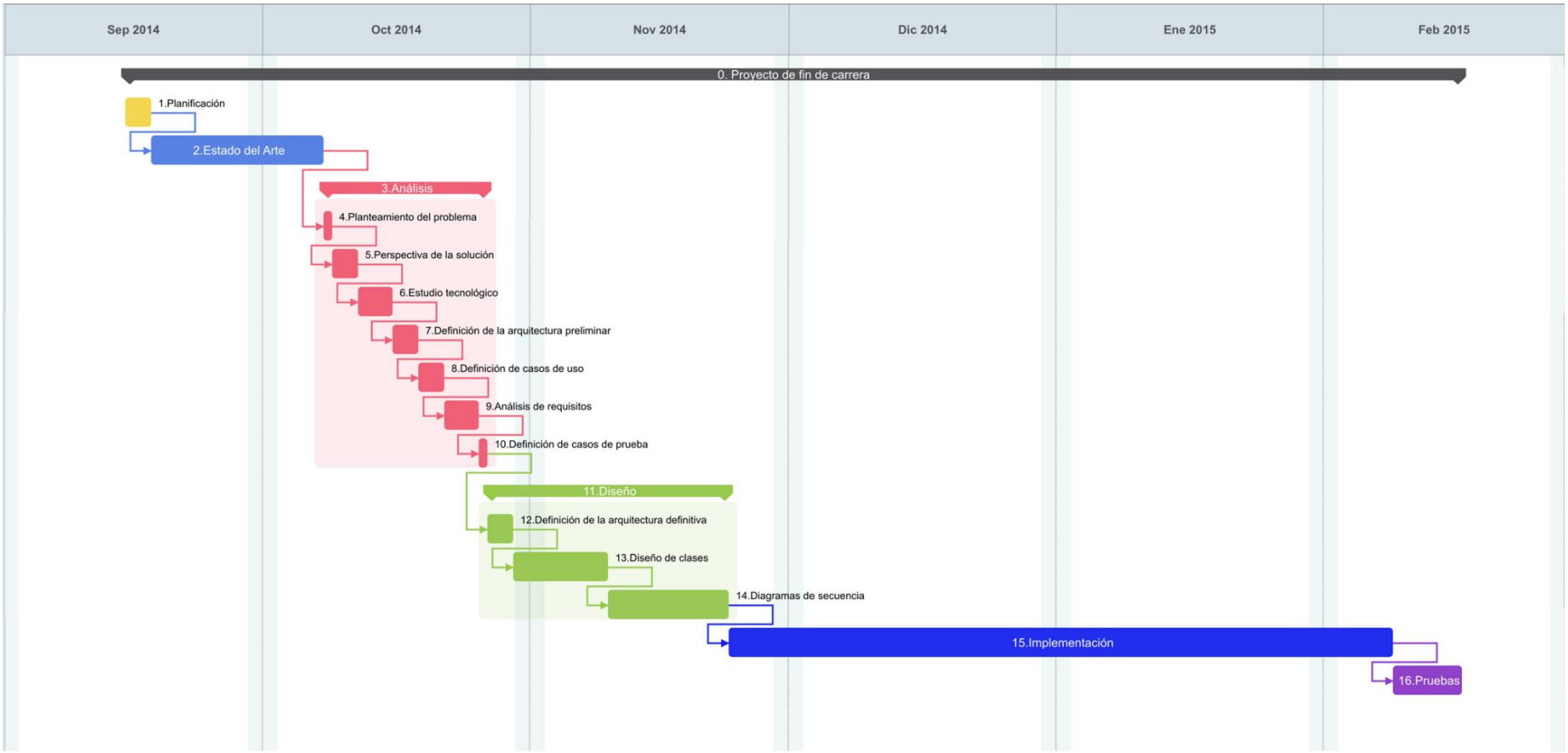

Tabla 20. Diagrama de Gantt de la planificación inicial

1.1.2 Desarrollo real del proyecto

En esta sección se muestra el desarrollo real del proyecto que posteriormente será comparado con la planificación inicial para estudiar las desviaciones surgidas a lo largo del proyecto.

En la planificación real se puede apreciar que el reparto de tiempos es similar al inicialmente estimado, si bien la duración de algunas de las tareas ha sido mayor. Durante el desarrollo del proyecto se incurrieron un total de 842 horas a lo largo de 191 días, habiéndose estimado inicialmente 685 horas repartidas en 155 días.

Nombre	Horas	Duración	Fecha de inicio	Fecha de fin
Proyecto	**842**	**191 días**	**15/09/2014**	**24/03/2014**
Planificación	**9**	**3 días**	**15/09/2014**	**17/09/2014**
Estado del Arte	**90**	**20 días**	**18/09/2014**	**07/10/2014**
Análisis	**98**	**23 días**	**08/10/2014**	**30/10/2014**
Planteamiento del problema	3	1 día	08/10/2014	08/10/2014
Perspectiva de la solución	14	3 días	09/10/2014	11/10/2014
Estudio tecnológico	23	6 días	12/10/2014	17/10/2014
Definición de la arquitectura preliminar	16	2 días	18/10/2014	19/10/2014
Definición de casos de uso	15	5 días	20/10/2014	24/10/2014
Análisis de requisitos	24	5 días	25/10/2014	29/10/2014
Definición de casos de prueba	3	1 día	30/10/2014	30/10/2014
Diseño	**143**	**31 días**	**31/10/2014**	**30/11/2014**
Definición de la arquitectura definitiva	19	3 días	31/10/2014	02/11/2014
Diseño de clases	62	14 días	03/11/2014	16/11/2014
Diagramas de secuencia	62	14 días	17/11/2014	30/11/2014
Implementación	**434**	**98 días**	**01/12/2014**	**08/03/2015**
Pruebas	**68**	**16 días**	**09/03/2014**	**24/03/2014**

Tabla 21. Planificación real del proyecto

Dado que el número de horas trabajadas no es el mismo para todos los días de la semana, es posible que una tarea cuya duración es, por ejemplo, de 3 días laborables no se empleen las mismas horas que para otra tarea de también 3 días de duración con un fin de

semana de por medio. Por este motivo, la comparativa y la desviación se calcula en términos de horas en vez de días.

Nombre	Planificado (horas)	Real (horas)	Diferencia (horas)	Desviación (%)
Proyecto	**685**	**842**	**160**	**22,92**
Planificación	**9**	**9**	**0**	**0,00**
Estado del Arte	**90**	**90**	**0**	**0,00**
Análisis	**87**	**98**	**11**	**12,64**
Planteamiento del problema	3	3	0	0,00
Perspectiva de la solución	14	14	0	0,00
Estudio tecnológico	17	23	6	35,29
Definición de la arquitectura preliminar	14	16	2	14,29
Definición de casos de uso	14	15	1	7,14
Análisis de requisitos	17	24	7	41,18
Definición de casos de prueba	8	3	-5	-62,50
Diseño	**124**	**143**	**22**	**15,32**
Definición de la arquitectura definitiva	9	19	10	111,11
Diseño de clases	53	62	12	16,98
Diagramas de secuencia	62	62	0	0,00
Implementación	**341**	**434**	**93**	**27,27**
Pruebas	**34**	**68**	**34**	**100,00**

Tabla 22. Análisis de la desviación del proyecto

Tal y como se puede observar en la tabla, la desviación total del proyecto es de un 22,92%. La tarea con mayor desviación fueron las pruebas, con un 100% de desviación. Esto es así debido a una subestimación del tiempo necesario para realizar las pruebas en la planificación inicial, ya que no se tuvo en cuenta que durante esta fase se podrían detectar errores en el código que tras solucionar, sería necesario volver a ejecutar el juego de pruebas.

La tarea de implementación es la que presenta la siguiente mayor desviación, con un 27,27%. Esto es debido principalmente a dos factores. Por un lado el desconocimiento inicial de algunas de las tecnologías presentes en el proyecto. Entre ellas cabe destacar por un lado el *framework* de *Spring*, que si bien es cierto que a la larga ahorra tiempo y trabajo al desarrollador, la curva de aprendizaje es elevada, y por otro la inexperiencia en el

desarrollo de extensiones para Joomla! La aparición de algunos obstáculos durante el desarrollo no contemplados inicialmente también ha tenido su impacto en el tiempo dedicado a la implementación.

Por último comentar que la tercera tarea con mayor desviación ha sido el diseño con un 15,32%. La principal variación viene dada por la sub-tarea de definición de la arquitectura definitiva, la cual fue necesario revisar en distintas ocasiones hasta dar con la arquitectura idónea para el sistema a desarrollar.

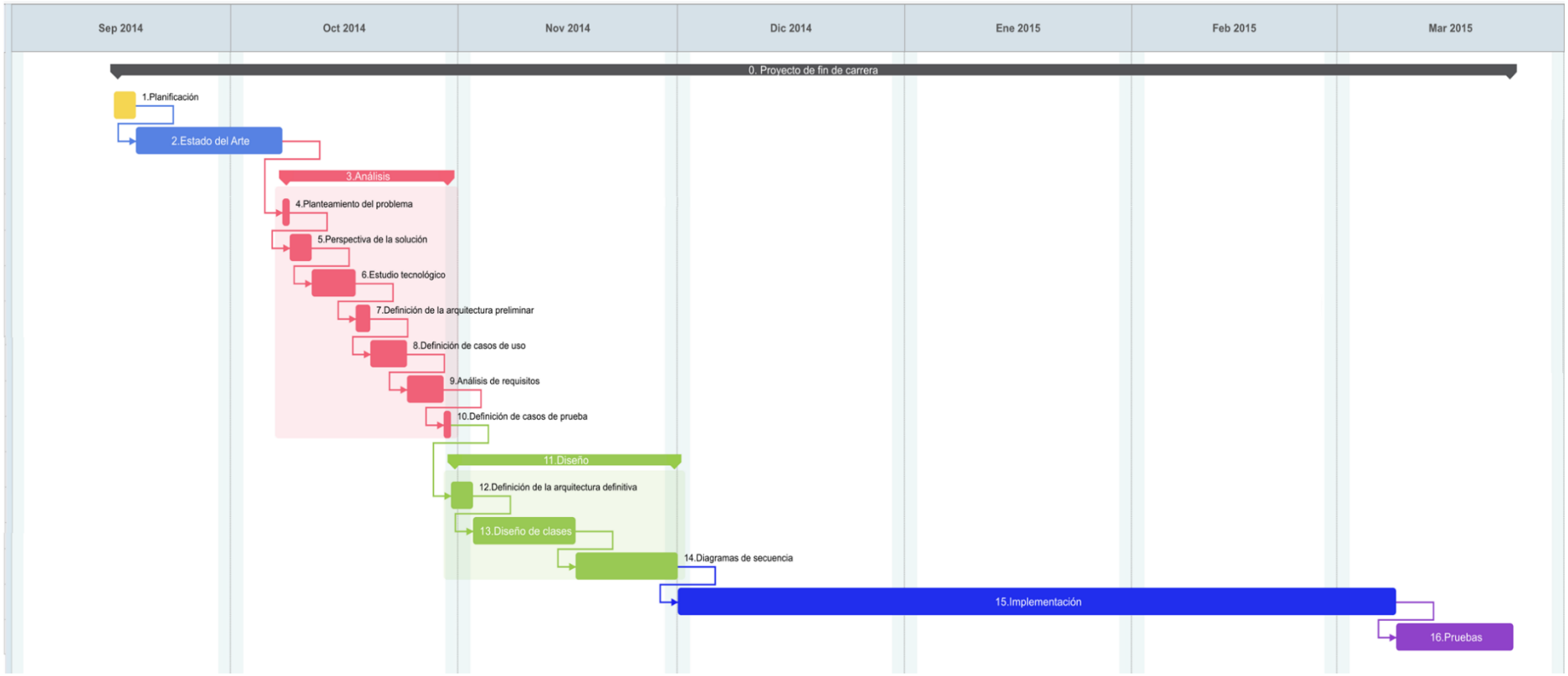

Figura 28. Diagrama de Gantt de la planificación real

1.2 Medios técnicos empleados para el proyecto

En esta sección se detallan las herramientas que se han utilizado a lo largo del desarrollo del proyecto. La tabla recoge las distintas herramientas junto con una breve descripción de cada una.

Herramienta	Descripción
Sistema Operativo Windows 7 Enterprise Edition (26)	Sistema operativo utilizado durante todo el desarrollo del proyecto.
Microsoft Office 2013 (27)	Conjunto de aplicaciones utilizadas para el desarrollo de la documentación.
Microsoft Project 2013 (28)	Aplicación utilizada para la creación de la planificación del proyecto.
Symantec Endpoint Protection (29)	Antivirus empleado para proteger el sistema operativo Windows 7.
Eclipse Kepler 4.3.2 Service Release 2 (30)	Eclipse es el entorno integrado de desarrollo empleado para la implementación de las partes que componen el sistema.
JEdit 5.1.0 (31)	Editor de textos utilizado para la visualización de ficheros de texto y código fuente.
Enterprise Architect Professional (32)	Sistema utilizado en el desarrollo de los diagramas asociados a las etapas de análisis y diseño.
WAMP Server (33)	Entorno de desarrollo que integra un servidor Apache, una base de datos MySQL y PHP. Se ha utilizado para la instalación de Joomla! y para alojar la base de datos del sistema desarrollado. Además incluye la herramienta PHPMyAdmin lo que ha facilitado la labor.
Ordenador portátil HP ProBook 430 G2	Ordenador portátil utilizado para todo el desarrollo del sistema y realización de la
iPhone 6 64GB	Teléfono móvil empleado como generador de claves OTP mediante la aplicación Google Authenticator.

Tabla 23. Medios técnicos empleados

1.3 Análisis económico del proyecto

En esta sección se lleva a cabo el análisis económico del proyecto. Primeramente se describe la metodología seguida, para pasar a detallar el presupuesto inicial estimado, así como el coste real del proyecto una vez finalizado.

Por otra parte, indicar que todas las operaciones están realizadas con una aproximación de dos dígitos decimales.

1.3.1 Metodología de estimación de costes

El presupuesto del proyecto se divide en dos partes, los costes directos y los costes indirectos:

Dentro de los costes directos, se incluyen aquellos elementos directamente implicados en el desarrollo del proyecto, tales como los recursos humanos, los equipos informáticos, el *software*, el material fungible, los viajes y las dietas en caso de apliquen.

Los costes indirectos, por su parte, se corresponden con aquellos gastos en los que se incurre necesariamente para el desarrollo del proyecto, pero que no están relacionados directamente con el desarrollo en sí mismo. Éstos incluyen el consumo energético, el coste del inmueble en el que se realiza el proyecto y el coste de la conexión a Internet. Para este proyecto, y de acuerdo con la plantilla proporcionada en la web de la Universidad, los costes indirectos se han calculado como un 20% de los costes directos.

1.4 Presupuesto inicial

En esta sección se presenta el presupuesto inicial donde se detallan los costes presupuestados para el proyecto junto con el presupuesto total estimado.

El 21% de IVA no está incluido en ninguno de los gastos reflejados en esta sección salvo en el coste total del proyecto en el que se indica expresamente que se ha contemplado el IVA para su cálculo.

1.4.1 Gastos de personal

La siguiente tabla detalla los gastos de personal. Para su cálculo se ha tenido en cuenta la dedicación de un único ingeniero a lo largo de todo el proyecto, con una dedicación de 3 horas diarias en días laborables y de 16 horas durante los fines de semana durante los 155 días de duración estimada del proyecto.

El coste hombre/mes del recurso se ha tomado del modelo proporcionado por la Universidad, siendo 1 hombre/mes equivalente a 131,25 horas de trabajo. Se ha considerado que este valor ya lleva incluido el importe destinado a pagar la Seguridad Social del trabajador, que conforma un 28,3% del coste total.

Categoría	Dedicación (hombres/mes)	Coste hombre/mes (€)	Coste TOTAL (€)
Ingeniero	5,22	2.694,39	**14.062,15**

Tabla 24. Gastos de personal

1.4.2 Gastos de equipos

En esta sección se detallan los gastos relacionados con los equipos utilizados a lo largo del proyecto. Dichos equipos son un ordenador portátil y un teléfono móvil. En la tabla muestra en detalle los costes imputables para cada equipo así como los datos utilizados en el cálculo.

El coste de los equipos es el precio de venta al público sin IVA. El periodo de depreciación para el ordenador es el utilizado en el ámbito de la Administración General mientras que el del teléfono es el periodo de lanzamiento entre nuevos modelos.

Descripción	Coste (€)	Dedicación (meses)	Periodo de depreciación (meses)	Coste imputable (€)
Ordenador portátil HP ProBook 430 G2 - Procesador Intel Core i5 4210U - 4 GB de memoria RAM - Disco duro de 128 GB SSD - Pantalla de 13.3”	724,99	5	36	100,69
iPhone 6 64GB	531,21	5	12	221,34
			TOTAL	**322,03**

Tabla 25. Gastos de equipos

1.4.3 Gastos de Software

En esta sección se detallan los gastos relativos al software utilizado a lo largo del proyecto. La tabla presenta el detalle de los costes imputables para cada aplicación de la misma forma que se hizo en la sección anterior con los gastos de equipos.

Esta tabla sólo muestra el software mencionado en la sección 1.2 de este anexo que no sea gratuito. El coste reflejado es el precio de venta al público sin IVA.

Adicionalmente se ha realizado una donación de 100 € para el desarrollo del software libre, ya que durante el desarrollo del proyecto se han utilizado diversas herramientas sin coste alguno que han sido de gran utilidad.

Descripción	Coste (€)	Dedicación (meses)	Periodo de depreciación (meses)	Coste imputable (€)
Sistema Operativo Windows 7 Enterprise Edition	285,00	5	36	39,60
Microsoft Office 2013 Professional	425,81	5	36	59,14
Microsoft Project 2013 Professional	1.081,51	5	36	150,21
Symantec Endpoint Protection	48,46	5	12	20,19
Enterprise Architect Professional	157,21	5	36	21,83
Donación Software libre	100,00	--	--	100,00
			TOTAL	**390,96**

Tabla 26. Gastos de software

1.4.4 Gastos de material fungible

En esta sección se detallan los gastos que se han realizado en concepto de consumibles. Dichos gastos se reducen para este proyecto a material de oficina tal como folios, cuadernos, bolígrafos, etc. De nuevo, el coste aquí reflejado no incluye el IVA.

Descripción	Coste unitario (€)	Cantidad	Coste imputable (€)
Material de oficina	25	1	25
		TOTAL	**25**

Tabla 27. Gastos de material fungible

1.4.5 Gastos de viajes y dietas

En esta sección se detallan los gastos de viajes y dietas realizados durante el proyecto.

Se ha considerado que a todos los efectos, el cliente es el tutor de este proyecto. Dado que el cliente está ubicado en Leganés, localidad fuera del término municipal de Madrid, los gastos en concepto de desplazamientos y dietas deben imputarse al proyecto.

Se ha viajado a Leganés por motivos de reunión con el cliente una vez al mes durante todo el desarrollo del proyecto. La distancia a las oficinas del cliente es de 27 Km, con un coste de 0.104 €/Km. Por cada desplazamiento se ha asignado una dieta de 8€/día.

Descripción	Coste unitario (€)	cantidad	Coste imputable (€)
Kilometraje	0,104 €/Km	270	28,08
Dietas	8	5	40
		TOTAL	**68,08**

Tabla 28. Gastos de viajes y dietas

1.4.6 Costes directos

En esta sección se muestran los costes directos asociados a este proyecto que son la suma de los conceptos calculados en los apartados anteriores: gastos de personal, gastos de equipos, gastos de software, gastos de consumibles y gastos de viajes y dietas.

En la tabla se muestra la suma de estos conceptos dando como resultado el total de costes directos del proyecto.

Descripción	Coste (€)
Gastos de personal	**14.062,15**
Gastos de equipos	**322,03**
Gastos de software	**390,96**
Gastos de consumibles	**25**
Gastos de viajes y dietas	**68,08**
TOTAL	**14.868,22**

Tabla 29. Costes directos

1.4.7 Costes indirectos

Los costes indirectos se han estimado como un 20% de los costes directos.

De esta forma y con los resultados obtenidos en la sección anterior, los costes indirectos del proyecto son **2.973,64 €**.

1.4.8 Estimación de costes

En la tabla inferior se detalla la estimación de costes del proyecto a partir de los cálculos realizados en las secciones anteriores. Tras la suma de todos los costes finalmente se aplica el IVA dando como resultado el total de los costes del proyecto.

Descripción	Coste (€)
Gastos de personal	14.062,15
Gastos de equipos	322,03
Gastos de software	390,96
Gastos de consumibles	25
Gastos de viajes y dietas	68,08
Costes directos	**14.868,22**
Costes indirectos	**2.973,64**
Total costes sin IVA	**17.841,86**
IVA (21%)	**3.746,79**
TOTAL	**21.588,65**

Tabla 30. Estimación de costes

1.5 Presupuesto para el cliente

En esta sección es muestra el presupuesto a presentar al cliente. Dada la magnitud del proyecto, este presupuesto consta de la suma de costes directos e indirectos añadiendo a mayores un porcentaje de riesgo con el que poder hacer frente a imprevistos surgidos durante el mismo. Finalmente se incorporan los beneficios esperados cuyo valor se calcula como un porcentaje del coste total incluyendo el riesgo.

En base a la experiencia en otros proyectos, el porcentaje de riesgo que se ha definido es del 15%. Por último, se añadirá un 20% más en concepto de beneficios por el desarrollo de la solución.

Descripción	Coste (€)
Gastos de personal	14.062,15
Gastos de equipos	322,03
Gastos de software	290,96
Gastos de consumibles	25
Gastos de viajes y dietas	68,08
Costes directos	**14.868,22**
Costes indirectos	**2.973,64**
Total Costes sin riesgo	**17.841,86**
Riesgo (15%)	2.676,27
Total Costes sin beneficios	**20.518,14**
Beneficios (20%)	4.103,63
Total costes sin IVA	**24.621,77**
IVA (21%)	5.170,57
TOTAL	**29.792,34**

Tabla 31. Presupuesto para el cliente

1.6 Coste final y análisis de la desviación

En este apartado se compara el coste real del proyecto con el coste estimado inicialmente. Debido al número de días adicionales que se ha incurrido en el desarrollo del proyecto, es de esperar una importante desviación en el presupuesto no sólo por los costes de personal sino también en las amortizaciones del material utilizado.

La tabla presenta una comparativa entre el presupuesto inicial y el coste final del proyecto, indicando las variaciones en cada concepto y el total. Para el cálculo del coste total de las distintas partidas del proyecto se han tomado los mismos costes base modificando los meses de desarrollo del proyecto así como el número de horas hombre para calcular así el coste real del proyecto.

Descripción	Coste presupuestado (€)	Coste real (€)	Diferencia (€)
Gastos de personal	14.062,15	17285,15	3.223,00
Gastos de equipos	322,03	402,54	80,51
Gastos de software	390,96	463,7	72,74
Gastos de consumibles	25	25	0,00
Gastos de viajes y dietas	68,08	81,7	13,62
Costes directos	**14.868,22**	**18.258,09**	**3.389,87**
Costes indirectos	**2.973,64**	**3.651,62**	**677,97**
TOTAL	**17.841,86**	**21.909,71**	**4.067,85**

Tabla 32. Coste final y análisis de la desviación

La diferencia entre el coste presupuestado y el coste real no se puede traducir en un aumento del precio a cobrar al cliente, por lo que se descontará del riesgo calculado. Teniendo en cuenta que el valor del riesgo es de 2.676,27 €, si descontamos la diferencia aún quedan 1.391,58 € que serán descontados del beneficio presupuestado. Finalmente se concluye que los beneficios finales del proyecto ascienden a 3.712,05 €.

Anexo II:

Manual de instalación del componente de autenticación de emergencia

2.1 Instalación del plug-in en el gestor de contenidos Joomla!

A continuación se detallan los pasos necesarios para la instalación del componente de autenticación de emergencia:

1. El primer paso de la instalación del *plug-in* consiste en copiar la carpeta "voucherLogin" del paquete de software entregado junto con el Proyecto de Fin de Carrera en la carpeta "\www\joomla_project\plugins" de la instalación de Joomla.

2. Luego, se deberá acceder a la consola de administración de Joomla!

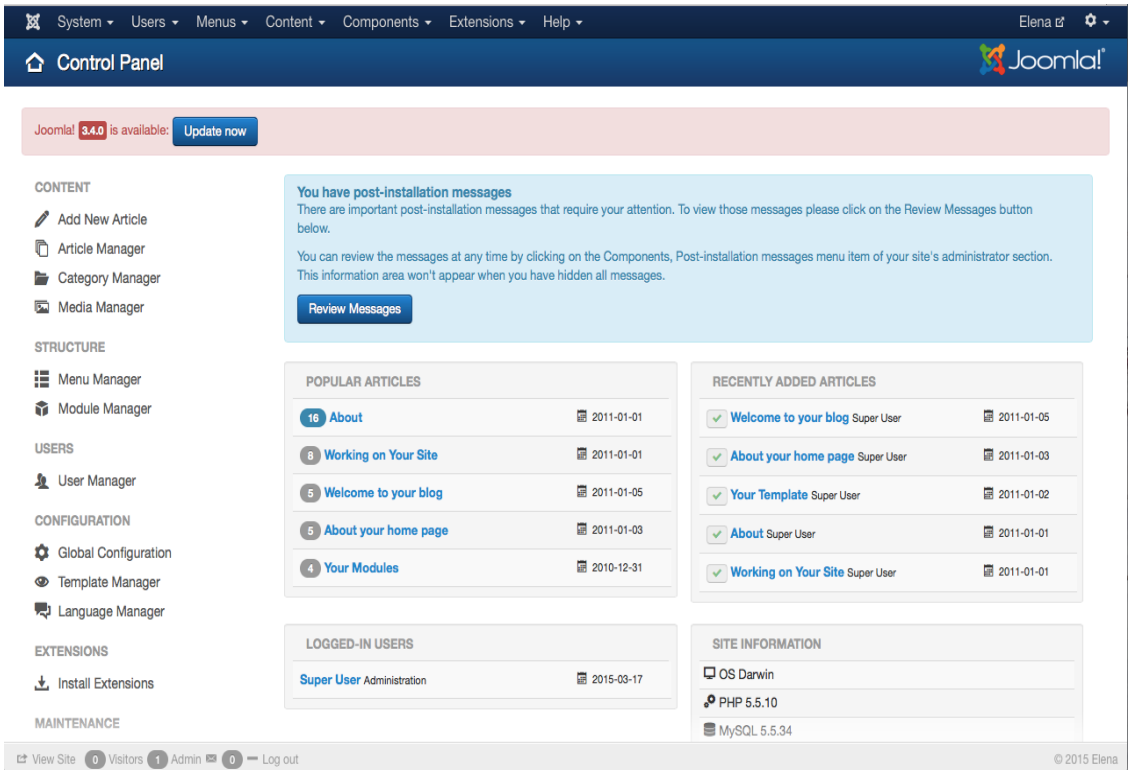

Figura 29. Paso 2 de la instalación del componente de autenticación de emergencia.

3. Una vez allí, se deberá acceder al menú “Extensions”, y seleccionar la opción “Extensions Manager”.

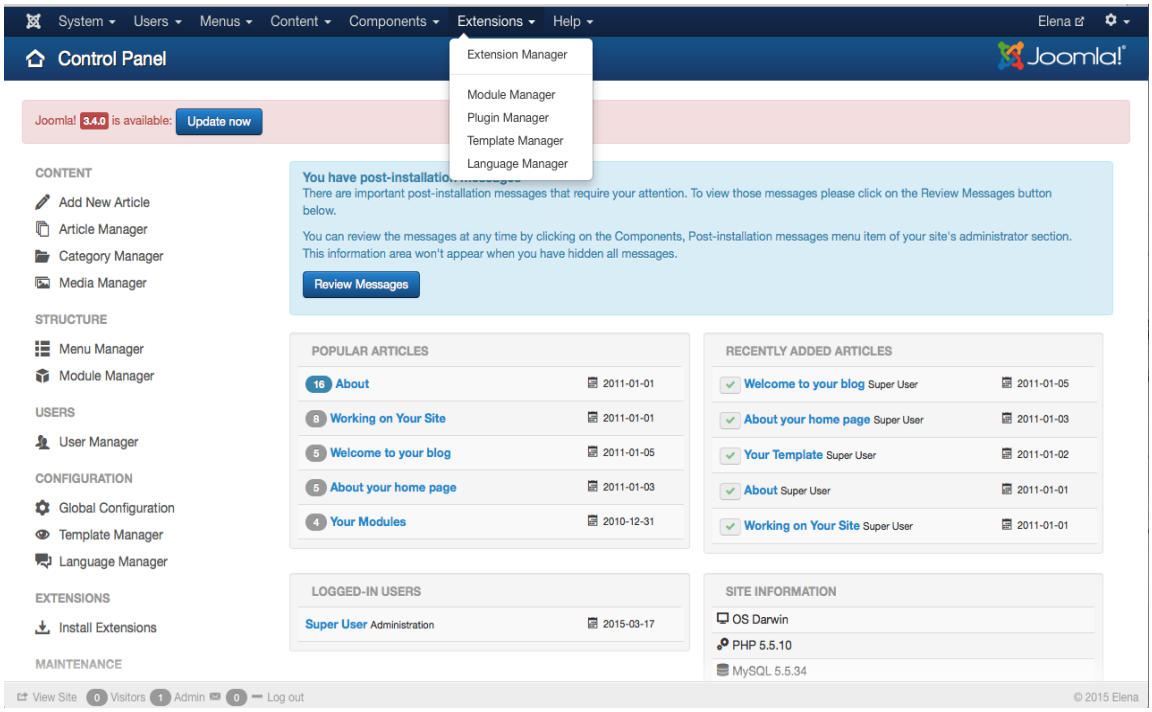

Figura 30. Paso 3 de la instalación del componente de autenticación de emergencia.

4. A continuación se deberá acceder al menú “Discover”.

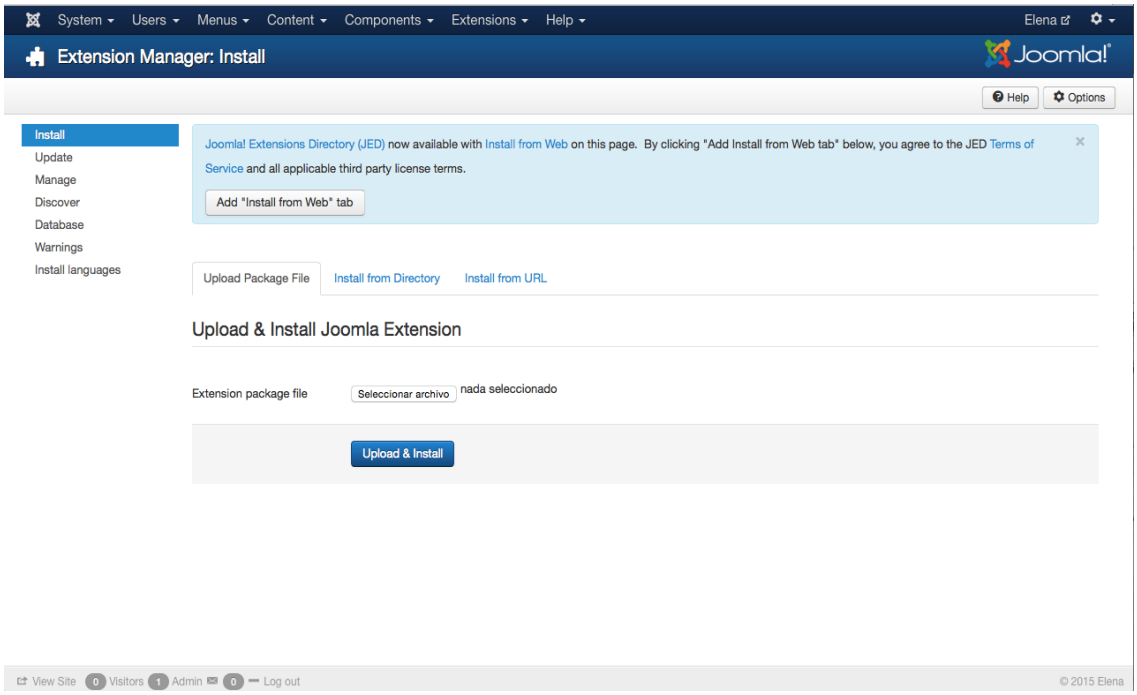

Figura 31. Paso 4 de la instalación del componente de autenticación de emergencia.

5. Allí deberá aparecer el *plugin* "voucherLogin".

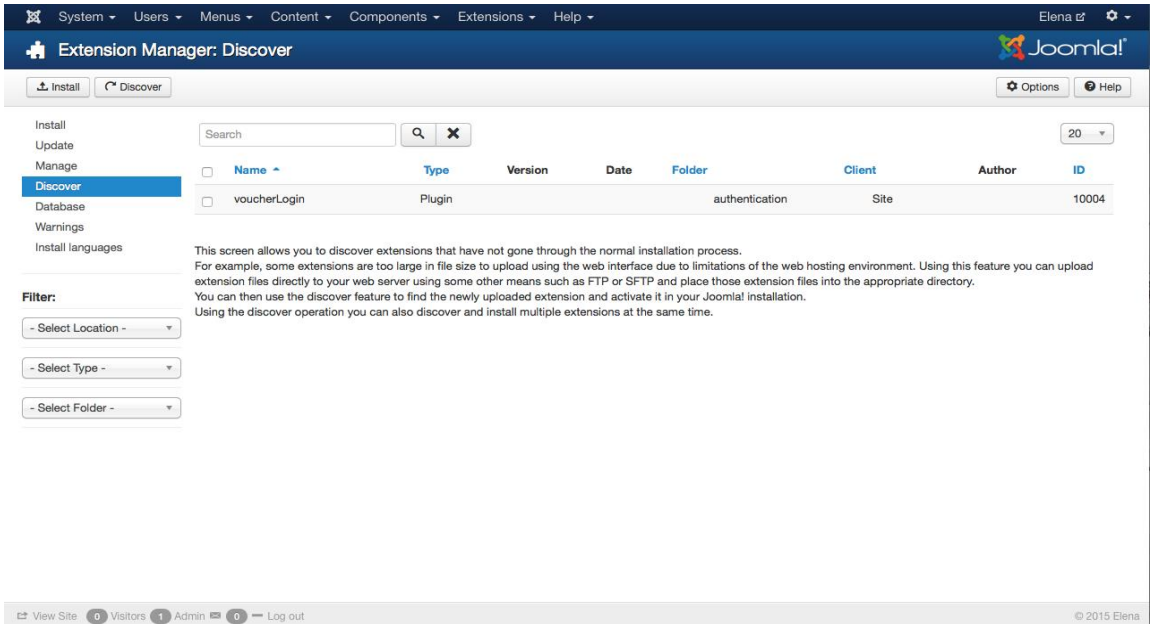

Figura 32. Paso 5 de la instalación del componente de autenticación de emergencia.

6. Se selecciona el *plugin* y se pulsa en instalar. Si todo ha ido bien, se mostrará un mensaje de confirmación.

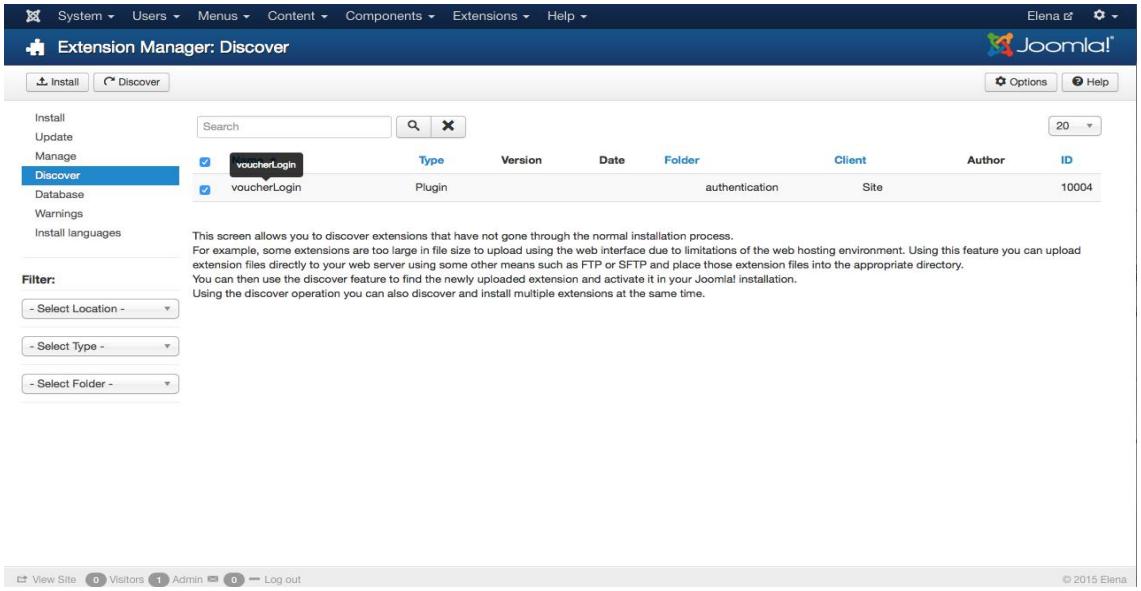

Figura 33. Paso 6 de la instalación del componente de autenticación de emergencia.

7. Finalmente, será necesario acceder de nuevo al menú "Extensions", "Plugin Manager" para activar el *plugin*.

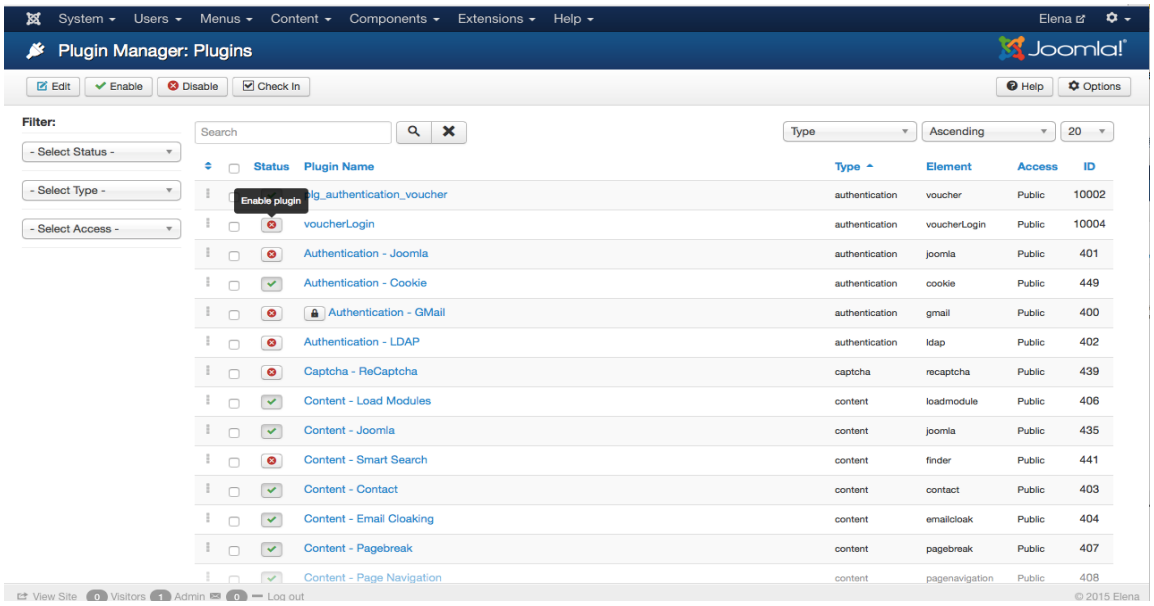

Figura 34. Paso 7 de la instalación del componente de autenticación de emergencia.